Escuela Politécnica Superior
Departamento de Tecnología Electrónica y de las Comunicaciones

APORTACIONES MEDIANTE IMPLEMENTACIÓN BASADA EN SISTEMAS EMBEBIDOS AL CONTROL DIGITAL DE CONVERTIDORES CONMUTADOS

TESIS DOCTORAL

Alberto Sánchez González

Madrid, junio de 2013

TÍTULO: Aportaciones mediante implementación basada en sistemas embebidos al control digital de convertidores conmutados

AUTOR: Alberto Sánchez González

DIRECTOR: Dr. Ángel de Castro Martín

El tribunal nombrado para juzgar la tesis doctoral antes citada, compuesto por los doctores:

PRESIDENTE: Dr. Óscar García Suárez

SECRETARIO: Dr. Javier Garrido Salas

VOCALES: Dr. Christian Brañas Reyes

Dr. Pablo Zumel Vaquero

Dr. Óscar Lucía Gil

acuerda otorgarle la calificación de

Madrid, a 20 de junio de 2013

A mi familia

Agradecimientos

Tres años pueden hacerse muy largos pero también muy cortos. En mi caso se han hecho extremadamente cortos y eso siempre es buena señal. El trabajo que ha supuesto la escritura de esta tesis doctoral está repleto de buenos momentos. He tenido la suerte, no tan habitual, de tener un director con grandes capacidades científicas pero siendo a la vez un amigo en el que confiar. Ya son bastantes años a sus órdenes y nunca he sentido que sea un jefe, sino un consejero que casi siempre acierta en sus intuiciones. Todo este trabajo ha sido posible gracias a su ayuda, tanto profesional como anímica. Él ha sabido paliar las carencias que yo tenía debido a la lejanía de esta tesis con mis estudios universitarios, y a la vez ha sabido explotar los conocimientos que sí tenía para aplicarlos a esta rama de conocimiento.

Si mirando atrás tengo este recuerdo tan agradable también es gracias al resto del grupo de investigación HCTLab. Quiero agradecer a Javier por darme esta oportunidad en el grupo. Entré por casualidad en el grupo y estar aquí se ha convertido en mi sueño profesional hecho realidad, así que sólo puedo agradecer su apoyo. No olvido los primeros momentos en el laboratorio, cuando Guillermo me *reclutó* hace tantos años. Han sido muchos proyectos con él y de todos guardo buen recuerdo. También quiero agradecer a Susana todos los buenos *momentos pumeros*, esas coca colas y esos atascos que hemos pasado juntos. Fernando también ha ayudado a despejar la mente con los cafés, las conversaciones y esos momentos de relax, tan necesarios cuando el trabajo aprieta y, como no, con sus ayudas con el LaTeX. También recuerdo esas conversaciones futboleras en la comida con Javi.

No quiero olvidarme de la gente de la Universidad de Cantabria. Me recibieron con los brazos abiertos y me hicieron sentir como en casa. En particular quiero agradecer a Paco esas clases teóricas que tanta falta me hacían. Y como no, a Víctor, Alejandro y Fran, que me trataron como a un amigo más, y que consiguieron que la estancia se hiciera muy corta.

Y por último, quiero mencionar a la gente más alejada de esta tesis en lo técnico, pero no por ello son menos importantes. A Mario, por todos esos momentos de desahogo y también por los de evasión. Y a Laura, por todas las experiencias vividas juntos durante estos años. Sabes desde hace mucho que eres una de las personas más importantes de mi vida.

Sólo tengo palabras de agradecimiento a mis padres, por darme el apoyo anímico y económico para estudiar lo que he deseado y por continuarlo durante estos primeros años de trabajo. Ellos también han sido fundamentales en este pequeño hito que es

la tesis doctoral. Y por último, no me olvido de mi hermano ya que siempre ha sido un modelo a seguir en los estudios y en lo profesional.

Resumen

En la última década, el control digital para convertidores conmutados de potencia ha evolucionado notablemente. Se ha demostrado que no sólo se pueden realizar las mismas tareas que en su vertiente analógica, sino que el control digital ofrece grandes ventajas. Esta tesis doctoral presenta por una parte un sistema para facilitar la etapa de pruebas del regulador digital para convertidores de potencia basándose en una arquitectura HIL (del inglés *Hardware In-the-Loop*), la cual emula el sistema completo de pruebas permitiendo grandes aceleraciones. Por otra parte, se muestra un método que realiza corrección de factor de potencia sensando únicamente la tensión de salida. Esta aproximación permite eliminar el sensado de la tensión de entrada y de la corriente de entrada del convertidor, siendo esta última especialmente significativa, debido a las desventajas que conlleva.

Una de las tareas imprescindibles en el desarrollo de un regulador digital es la etapa de pruebas, especialmente en sistemas donde un fallo del regulador sea crítico. Esta etapa, sin embargo, no es trivial dado que la naturaleza del convertidor es analógica, mientras que la del regulador es digital. No sólo es útil la simulación de un modelo simplificado del regulador junto a un modelo de la planta, sino que es deseable simular la implementación real del regulador junto a un modelo de la planta. En el caso de reguladores escritos en HDL (del inglés *Hardware Description Language*), estos pueden ser simulados junto a un modelo HDL de la planta, creando una simulación íntegramente digital, siendo mucho más rápida que una simulación mixta analógica-digital. Sin embargo, hasta las simulaciones digitales pueden llegar a ser extremadamente largas dependiendo de la aplicación que se desee simular, como puede ser la corrección de factor de potencia. Por tanto, puede que las simulaciones digitales no sean útiles en cuanto al tiempo de simulación, por lo que surge la necesidad de realizar un sistema HIL, es decir, emular el sistema completo, realizando las pruebas en *hardware* real y en tiempo real. El modelo digital de la planta puede realizarse con diferentes aritméticas, las cuales repercuten enormemente en el tiempo de simulación e incluso en la precisión de la simulación. En la presente tesis doctoral se muestra la metodología para implementar el modelo de una planta usando diversas aritméticas que implementan coma flotante y coma fija, y se profundiza en la resolución numérica de los modelos presentados. Asimismo, se presenta un estudio exhaustivo que compara todas las aritméticas presentadas y se demuestra que los resultados obtenidos tienen gran similitud, y por tanto utilidad, con los resultados experimentales que se obtienen con el convertidor real.

La segunda parte de la tesis doctoral presenta aportaciones relacionadas con la corrección de factor de potencia realizada de forma digital. En las técnicas tradicionales deben utilizarse tres ADCs (del inglés *Analog-to-Digital Converter*) para medir las tensiones de entrada y de salida, así como la corriente de entrada del convertidor. Aprovechando las ventajas del control digital se propone precalcular el ciclo de trabajo del conmutador del convertidor, y aplicarlo posteriormente, aprovechando la naturaleza periódica de la corrección de factor de potencia. Idealmente se necesitaría únicamente la sincronización con la red eléctrica, la cual es posible usando un comparador analógico de tensión. Dado que las condiciones reales de operación varían respecto a las ideales, debe realizarse una modificación en tiempo real del ciclo de trabajo precalculado. Para ello se muestran diversas técnicas las cuales dividen el ciclo de trabajo en diferentes componentes que pueden ser tratadas de forma diferente para poder adaptarse a tensiones de entrada y potencias no nominales. Todas las técnicas propuestas utilizan un único ADC que mide la tensión de salida. La medida del ADC se utiliza para calcular la tensión media de salida y su rizado, que a su vez depende de la carga, y estos se utilizan en las técnicas de control. Por tanto, todas las técnicas propuestas utilizan un comparador de tensión para la sincronización y un ADC, frente a las técnicas clásicas que usan tres ADCs. Los resultados experimentales demuestran que las técnicas presentadas cumplen la normativa IEC-61000-3-2 en condiciones nominales y frente a variaciones notables en la tensión de entrada y carga del convertidor.

Abstract

During the last decade, digital control for switching power converters has evolved considerably. It has been shown that digital control can perform not only the same tasks as analog control, but digital control offers big advantages. On the one hand, this thesis presents a system to improve the testing stage of digital controllers for power converters based on an HIL (Hardware In-the-Loop) architecture, which emulates the whole test system allowing high acceleration. On the other hand, it shows a method for PFC (Power Factor Correction) which only senses the output voltage of the converter. This approach does not sense the input voltage and input current of the converter, the latter being particularly significant because of the disadvantages which it implies.

One of the essential tasks in the development of a digital controller is the testing stage, especially when a failure of the regulator is critical. However, this stage is not trivial because the converter is analog, while the regulator is digital. The simulation of a simplified model of the regulator is useful, but it would be desirable to simulate the real implementation of the controller with a model of the plant. In the case of HDL (Hardware Description Language) regulators, they can be simulated with a HDL model of the plant, creating a full-digital simulation, which is much faster than a mixed analog-digital simulation. Nevertheless, even digital simulations can be extremely long depending on the application that must be simulated, e.g. power factor correction. Therefore, digital simulations may not be useful in terms of simulation time, so an HIL system is needed, i.e. to emulate the entire system, performing the test in real hardware and in real-time. The digital model of the plant can be made with different arithmetics, which greatly affect the simulation time and even the simulation accuracy. This thesis shows the methodology to implement the model of a plant using different floating point arithmetics and also fixed point arithmetics, and it delves into the numerical resolution of the presented models. It also presents a thorough study that compares all the presented arithmetics and it is shown that all the results are very similar to experimental results taken with a real converter.

The second part of the thesis presents contributions related to power factor correction using digital control. Traditional techniques use three ADCs (Analog-to-Digital Converter) to measure the input and output voltages and the input current of the converter. Taking advantage of the digital control it is proposed to pre-calculate the switching duty cycle of the converter and apply it to the switch, as the power factor correction is a periodic task. Ideally it only requires synchronization with the mains, which is possible to reach using an analog voltage comparator. Since actual

operating conditions vary with respect to the ideal ones, a real-time modification of precalculated duty cycle should be performed. Several techniques are proposed, all of them dividing the duty cycle in several components that are differently treated in order to compensate for changes in the input voltage or in the load of the converter. All the proposed techniques use only one ADC, which measures the output voltage. This ADC measure is used to calculate the average output voltage and the output voltage ripple, which depends on the load, and these are used in several control loops. Therefore, all the proposed techniques use only one voltage comparator for synchronization with the mains and one ADC, while conventional techniques use three ADCs. The experimental results show that the techniques presented meet the IEC-61000-3-2 regulation at nominal conditions even with substantial changes in the input voltage and the load.

Índice general

Apéndices

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y motivación

El uso de controladores digitales para el control de convertidores conmutados ha pasado de ser únicamente un campo de investigación a ser una realidad comercial. Aun así, el uso de controladores analógicos es todavía mayoritario debido a ciertos inconvenientes del control digital, como el coste o la dificultad de su desarrollo, que siguen siendo objeto de investigación. Es ahí precisamente donde se centra esta tesis doctoral. Por otro lado, las ventajas del control digital son claras. Se ha demostrado que el control digital no sólo iguala las prestaciones del control analógico, sino que lo supera en muchos aspectos. El control digital permite aumentar las prestaciones de la regulación, añadir nuevas funcionalidades e incluso reducir costes del convertidor. Por ello no es de extrañar que su uso esté creciendo continuamente.

Como se ha comentado, una de las dificultades para la implantación del control digital está en su desarrollo, y en concreto en la depuración de un sistema mixto analógico (planta) y digital (control), como muestra la figura 1.1. Por supuesto, los controladores de potencia deben ser ampliamente probados antes de ser implementados en un convertidor real. Esto es debido a que cualquier error en el control puede

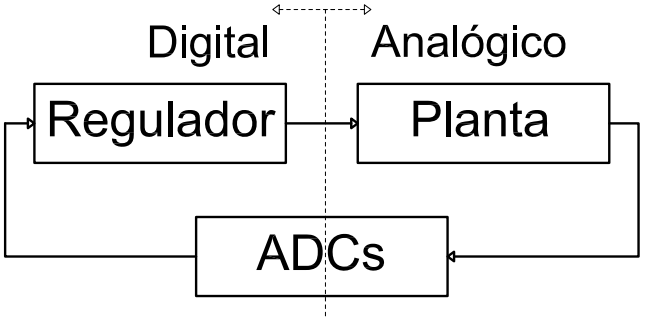
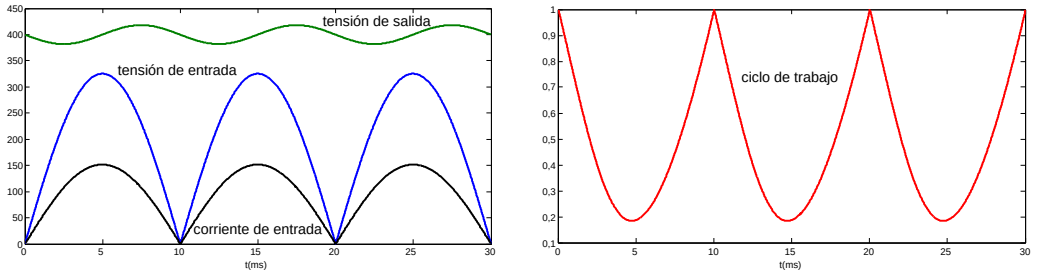


Figura 1.1: Naturaleza analógica-digital en un convertidor de potencia controlado digitalmente

provocar daños irreparables al convertidor. La simulación de reguladores analógicos se puede hacer fácilmente con un programa de simulación analógico, de tipo SPICE. Sin embargo, en el control digital, la simulación no es una tarea trivial, ya que el sistema completo tendrá una etapa digital, pero también una etapa analógica, que es la etapa de potencia.

La primera parte de esta tesis doctoral se basa en la depuración conjunta de ambas partes, digital y analógica, de convertidores controlados digitalmente. En particular, el capítulo 2 presenta diferentes formas de modelar el convertidor de potencia en un lenguaje de descripción de *hardware* o HDL. De esa forma, tanto el regulador escrito en un lenguaje HDL como la planta pueden simularse de forma conjunta, y cualquier error puede ser detectado. Esta simulación conjunta se conoce como HIL. La gran ventaja de esta simulación es que el mismo regulador que se implementará en la versión final es el que se simula, por lo que los resultados de la simulación son fiables para la etapa de pruebas del regulador. Una forma de realizar la simulación de todo el sistema es mediante un simulador mixto analógico-digital pero, como se muestra en el capítulo, las simulaciones resultantes pueden llegar a ser extremadamente largas, haciendo que la etapa de pruebas sea imposible de abordar dependiendo de la aplicación a simular. El modelado del convertidor en HDL permite realizar simulaciones mucho más cortas, y además permite la emulación en *hardware* de todo el sistema, por lo que la etapa de pruebas se puede realizar a grandes velocidades. El capítulo muestra una comparación exhaustiva de diferentes modelos HDL de la planta, por lo que proporciona a futuros diseñadores diferentes alternativas para modelar otros convertidores en función de las necesidades requeridas: tiempo de simulación, resolución de la simulación, facilidad de diseño, etc. Además, se han incluido pérdidas eléctricas a los modelos para mostrar en qué medida son influyentes en la simulación, y se han comparado con resultados experimentales, y con métodos tradicionales de simulación, como la simulación mixta analógica-digital. El ejemplo de aplicación para esta primera parte es la simulación de un regulador para corrección de factor de potencia. Esta aplicación es especialmente adecuada, ya que las simulaciones requeridas son largas. En corrección de factor de potencia la frecuencia de conmutación puede ser alta, en torno a cientos de kiloherzios, mientras que la dinámica de la planta puede estar en torno a décimas de segundo. Además el reloj del controlador puede estar en torno a los megaherzios, por lo que la simulación resultante es muy extensa. Aunque el capítulo muestra esta aplicación, cualquier otro convertidor puede ser simulado siguiendo los conceptos explicados durante el capítulo.

Aparte de la dificultad de diseño, el otro gran factor que limita la implantación del control digital es el precio. Por eso es habitual que se busque con el control digital alguna ventaja que permita reducir el precio del sistema en su conjunto. Ahí es



(a) Tensiones de entrada y salida, y corriente de entrada.

(b) Ciclo de trabajo.

Figura 1.2: Formas de onda en la corrección de factor de potencia.

donde se centra la segunda parte de esta tesis doctoral, y en concreto en la aplicación de corrección de factor de potencia. Las técnicas clásicas de corrección factor de potencia requieren la cuantificación de las tensiones de entrada y de salida, y de la corriente de entrada, requiriendo tres ADCs. Cada medición incrementa el coste el regulador final, por lo que sería deseable evitarlas. Además, la corriente de entrada es especialmente compleja, ya que su medición suele suponer pérdidas eléctricas si ésta se hace mediante un sensor resistivo, o la precisión, tamaño y coste no son las ideales. Aprovechando las ventajas del control digital, el capítulo 3 presenta un método de corrección de factor de potencia en el que los ciclos de trabajo que se aplican al interruptor son precalculados, en vez de calcularse en tiempo real midiendo las tres señales analógicas descritas. Esto es posible ya que las formas de onda que se deben tener en cuenta en la corrección de factor de potencia (tensión de entrada, tensión de salida, y corriente de entrada), así como el ciclo de trabajo, son periódicas, como muestra la figura 1.2. El precálculo del ciclo de trabajo se realiza una única vez en un ordenador, mientras que el regulador lo lee de una memoria, y lo aplica. A priori, el único requisito del controlador es una memoria y sincronización con la red eléctrica, ya que ésta ofrece corriente alterna. Esta sincronización puede realizarse con un simple comparador de tensión de bajo ancho de banda, en vez del ADC usado en las técnicas clásicas, por lo que el coste se reduce. Dado que el ciclo de trabajo precalculado es válido para unas condiciones de trabajo concretas, el controlador necesita modificarlo para adaptarse a las nuevas condiciones. Para ello el capítulo muestra métodos con los que el controlador adapta los ciclos de trabajo ante cambios de la tensión de entrada y de potencia usando un único ADC para medir la tensión de salida. El capítulo presenta diferentes métodos para dichas regulaciones, todas ellas usando un único ADC, y la sincronización con la red eléctrica usando un comparador de tensión.

Si bien los dos temas de esta tesis doctoral pueden parecer un tanto independientes,

y de hecho se pueden aplicar por separado, los resultados de la primera parte (depuración) se han aplicado durante el desarrollo de la segunda parte (corrección de factor de potencia) por tratarse precisamente de una aplicación que requiere simulaciones especialmente largas. La estructura de la presente tesis doctoral es la siguiente:

- El capítulo 2 muestra los diferentes métodos para la simulación y emulación de un sistema mixto analógico-digital.
- El capítulo 3 explica el sistema para corrección de factor de potencia en el que se precalcula el ciclo de trabajo y éste es regulado posteriormente usando un único ADC.
- Las conclusiones finales de la tesis doctoral se muestran en el capítulo 4.
- Por último, el apéndice A muestra un listado de códigos usados para la simulación y emulación HIL, mientras que el apéndice B presenta un glosario de abreviaturas usadas durante el documento.

Capítulo 2

Verificación de controladores digitales

2.1. Introducción

No hay duda de la importancia de la depuración de reguladores para convertidores conmutados ya que, al tratarse de fuentes de alimentación, se maneja una potencia no despreciable. Si el diseñador prueba su regulador directamente con el convertidor real, sin depuración previa, la prueba puede ocasionar daños materiales o incluso personales. Cuando el regulador es digital, el proceso de depuración es más complejo porque se debe simular un sistema mixto analógico-digital.

Este capítulo se centra en la depuración de reguladores digitales diseñados en un lenguaje de descripción de *hardware* (HDL), que es la elección más común cuando se van a usar FPGAs (del inglés *Field-Programmable Gate Array*) o ASICs (del inglés *Application Specific Integrated Circuits*). Aunque los conceptos explicados en este capítulo son aplicados al lenguaje VHDL, la mayor parte de ellos puede ser directamente aplicados a otros lenguajes como Verilog HDL. Adicionalmente, algunos conceptos son aplicables para regulares o modelos de plantas no descritos en lenguajes HDL, tales como reguladores implementados en un lenguaje *software*, como puede ser C. En particular, para este tipo de lenguajes de programación, son válidos los conceptos de precisión y resolución de las variables.

Una vez que se ha diseñado la función de transferencia o algoritmo de control de un regulador digital para un convertidor de potencia, éste debe ser implementado en *hardware*. Sin embargo, la descripción en un lenguaje HDL puede generar errores en el sistema. Estos errores pueden ser debidos a codificación errónea o por detalles

de la implementación, tales como el rango y resolución de las señales numéricas, segmentación, sincronización, etc.

El regulador seguramente se habrá diseñado y probado usando herramientas informáticas, como puede ser Matlab [1] o SISO Design Tool, ambas de la compañía The MathWorks. Por tanto, las simulaciones que se describen aquí no comprueban solamente la estabilidad del controlador, ancho de banda, etc. Los sistemas de simulación que se proponen aquí se deben utilizar tras la descripción en HDL, es decir, en su **estado de codificación final**. Esta fase es importante para evitar daños en el equipo o incluso personales. El regulador final, que se implementará en *hardware*, debe ser probado junto a un modelo del convertidor de potencia. La dificultad de esta simulación se debe a que el regulador es un componente digital y está descrito mediante un lenguaje HDL, mientras que el convertidor es un componente analógico.

En este capítulo se muestran diferentes posibilidades para modelar un convertidor de potencia y simularlo junto a un regulador digital. En particular, se presentan modelos descritos directamente en esquemáticos y en VHDL, y dentro de este lenguaje, se describen modelos usando señales de tipo coma flotante simulable y emulable y coma fija. Igualmente se muestra al final del capítulo una comparativa entre todos los sistemas de simulación y emulación, resaltando las características principales de cada uno. A modo de comparativa, se añaden resultados experimentales realizados con un prototipo real. De esta forma, se demostrará la gran similitud entre los modelos simulados o emulados y el comportamiento de un convertidor real. Por último, el capítulo muestra un estudio heurístico sobre la resolución que deben tener las señales del modelo, y una batería de pruebas automatizada para comprobar la validez de dicho estudio.

2.1.1. Estado del arte

La simulación de sistemas mixtos, esto es, sistemas que combinan componentes digitales y analógicos, no es un problema nuevo, sino que muchas propuestas han sido presentadas con anterioridad.

Una de las primeras propuestas [2] analizó cuatro modelos de un regulador para un convertidor reductor (o *buck* en inglés). Dos de los modelos usan Matlab y Simulink analizando el sistema completo (regulador, convertidor y ADCs) en el dominio de la frecuencia y del tiempo. Esta aproximación permite añadir no idealidades como retrasos, ciclos de trabajo límites para la señal PWM (*Pulse-Width-Modulation*) de control del mosfet, etc. Sin embargo, el mayor problema de esta aproximación es que el regulador que se implementará finalmente en *hardware* no es el mismo que

el diseñado en la Modelsim/Simulink. El tercer modelo que comparan consiste en la creación de un modelo del regulador en Verilog HDL mientras que el convertidor sigue siendo un componente digital. En este caso se propone utilizar el simulador mixto Cadence Spectre [3] y se obtienen tiempos mayores de simulación. Por último proponen la simulación a nivel de dispositivo tanto para el regulador como para el convertidor con modelos Spice, obteniendo simulaciones de horas para un ciclo de conmutación.

Solamente hay unos pocos simuladores que soporten simulaciones mixtas analógica-digitales, por lo que algunas propuestas han descrito simulaciones usando dos simuladores [4], uno para la parte analógica (PSIM) y otro para la parte digital (Modelsim [5]). El mayor problema de esta solución es que el ingeniero de pruebas debe crear específicamente un enlace entre ambas aplicaciones usando lenguajes de programación, tales como C/C++, figura 2.1. A la hora de diseñar estas interfaces, se debe tener en cuenta la sincronización de datos, el flujo de re-alimentación entre los simuladores, etc.

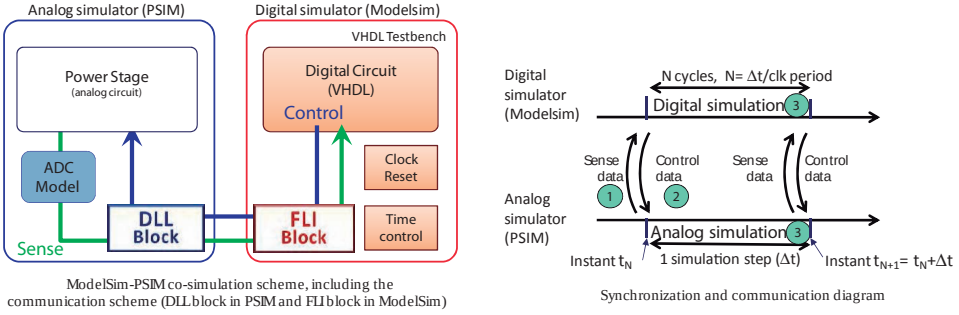


Figura 2.1: Co-simulación PSIM-Modelsim. Imagen extraída de [4].

Otra posibilidad es modelar el convertidor de potencia en HDL implementando ecuaciones en diferencias que describen el convertidor, lo que permite simulaciones más rápidas [6, 7]. Sin embargo, este método requiere que el ingeniero codifique a mano el modelo del convertidor de potencia. En la propuesta [8] se comparan diferentes modelos de un convertidor reductor, usando Spice, VHDL-AMS (una extensión de VHDL para definir señales analógicas y digitales simultáneamente) comportamental y VHDL. En comparación con Spice, la simulación VHDL-AMS es 2,18 veces más rápida, y la simulación VHDL es 3, 85 veces más rápida. Otra comparación entre VHDL y VHDL-AMS se puede encontrar en [9], consiguiendo tiempos de simulación 10 veces menores en la simulación VHDL. En [10] proponen una metodología de simulación mixta con diferentes niveles de abstracción para probar reguladores de convertidores de potencia. En particular, proponen el diseño del sistema en VHDL-AMS, mientras que el regulador está escrito en VHDL. En [11] se utiliza un simulador mixto (figura

2.2) para un sistema cuyo convertidor está descrito en Spice, el regulador en VHDL sintetizable y los ADCs en VHDL-AMS.

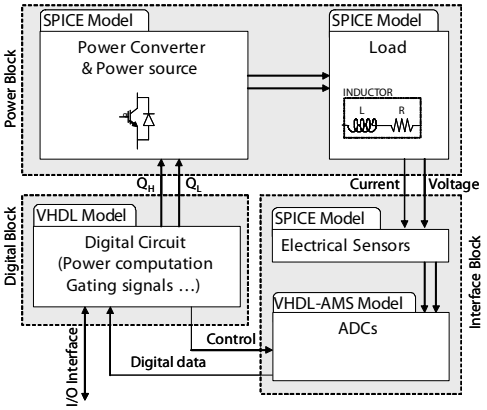


Figura 2.2: Entorno de simulación mixta.

Las simulaciones con modelos HDL del convertidor de potencia son más rápidas que las simulaciones analógicas, pero puede que no sean suficientemente rápidas para simulaciones complejas. Una aplicación donde es deseable tener técnicas más rápidas es la aplicación de corrección de factor de potencia, en la que se requieren simulaciones de cientos de milisegundos. Además, hay reguladores complejos cuya simulación es extremadamente lenta, por ejemplo aquellos que usan procesadores embebidos. El código de un regulador ejecutado en un procesador puede depurarse vía *software*, pero si se quiere probar el sistema final, se debe realizar una simulación tanto de los componentes analógicos, como de la estructura interna del procesador ejecutando el programa del regulador. Dado que el modelo de un procesador es muy complejo, esta simulación requiere de horas para simular pocas líneas de código. En [12, 13, 14] se muestra cómo emular convertidores para cocinas de inducción utilizando reguladores *software*. El sistema propuesto cuenta con un regulador que se ejecuta en un microprocesador embebido en una FPGA (figura 2.3).

Cuando la simulación es muy larga, una solución es usar un sistema HIL (*Hardware In-the-Loop*). En un sistema HIL, el modelo del convertidor es implementado en *hardware* digital (un ordenador, un microprocesador o una FPGA) para emular todo el sistema en lazo cerrado. Las primeras propuestas sobre HIL utilizaron ordenadores [15, 16], pero el paso de integración del modelo estaba sobre los cientos de μs, así que solamente podía aplicarse a sistemas de baja frecuencia de conmutación (menores a 10 kHz). El paso de integración indica cada cuánto tiempo del sistema simulado se calculan nuevos valores para sus variables, siendo más preciso el cálculo cuanto menor sea este tiempo. En [16] se aplicaron técnicas *software* de tiempo real basadas en Linux para reducir el tiempo de integración hasta 50 μs, aunque estando lejos de

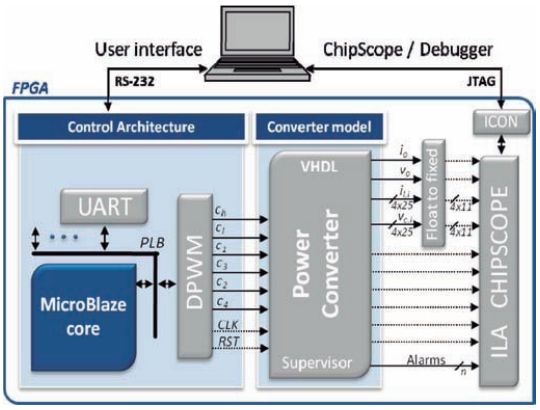


Figura 2.3: Emulación de un sistema con procesador embebido.

sistema de media-alta frecuencia de conmutación (mayores de 100 kHz). En otros campos también se ha visto la relación entre la resolución y la precisión de una simulación y la necesidad de llegar a un compromiso entre precisión y rapidez de la simulación [17].

Para poder reducir aún más el tiempo de integración (hasta decenas o centenas de ns), y así poder probar reguladores de media-alta frecuencia de conmutación, se pueden usar FPGAs o DSPs (*Digital Signal Processor*). Se han presentado varias propuestas de sistemas HIL en FPGAs [18, 19, 20, 21, 22], así como sistemas basados en DSPs [23, 24]. Aún así, en todos los casos previos se presentaron modelos de convertidores con baja frecuencia de conmutación, con lo que no habían surgido los problema de resolución que se presentarán en este capítulo. En particular, las propuestas [19, 20, 21, 22] usan modelos del convertidor en coma fija. Los modelos en coma fija obtienen los mejores resultados en cuanto a la velocidad de simulación, pero implican un mayor esfuerzo de diseño. De hecho, en las propuestas [18, 21, 22] se usó un modelo realizado en Matlab traduciéndolo a VHDL con herramientas automáticas. Esta conversión automática es cómoda para el diseñador pero hasta hace poco no generaba soluciones sintetizables. Incluso siendo sintetizables, esta solución no ofrece las mismas prestaciones en cuanto a frecuencia de trabajo ni área ocupada en la implementación.

En [13, 14] se utiliza el paquete VHDL2008 *float_pkg* [25] en un sistema HIL, utilizando señales en coma flotante, y se implementa en *hardware*. En estos casos, el regulador usa un procesador embebido Microblaze, por lo que una simulación es inviable, así que se opta por la emulación. El uso de coma flotante mediante esta biblioteca permite realizar el modelo del convertidor de forma sencilla, ya que solamente hay que codificar directamente las ecuaciones en diferencias que definen el convertidor de potencia utilizado, sin tener en cuenta anchos y formatos de cada señal.

Sin embargo, el mayor problema de esta biblioteca es que solamente es soportada por unas pocas herramientas de síntesis, así que no siempre se puede utilizar para técnicas de HIL. Una desventaja adicional de esta biblioteca es que, como se verá, necesita una gran cantidad de recursos *hardware* y su frecuencia de funcionamiento, en el caso de síntesis, es mucho inferior al uso de coma fija.

Este capítulo presenta el desarrollo de diferentes modelos de un convertidor de potencia usando las principales técnicas descritas en el estado del arte, y presenta una comparación cualitativa y cuantitativa de todas ellas. Además, el capítulo se centra especialmente en los modelos desarrollados en coma fija, aportando optimizaciones en frecuencia y área del *hardware* resultante para emulación.

2.2. Posibilidades de simulación

Como se ha comentado en la introducción, no sólo debe ser probada la función de transferencia del regulador digital para un convertidor de potencia. Cuando se codifica un regulador en un lenguaje de descripción de *hardware*, se pueden producir errores de codificación. Además, se producen no idealidades, como limitación del mínimo y máximo ciclo de trabajo del PWM, retrasos en los ADCs, etc. Debido a estas razones, es crítico probar el regulador en su estado de codificación final junto a un modelo del convertidor de potencia. Dado que el regulador es un componente digital y el convertidor es analógico, la simulación no es trivial, sino que requiere una simulación más compleja. El modelo del convertidor determina el tipo de simulación que puede ser realizada.

Una posibilidad es utilizar un **simulador mixto analógico-digital**, el cual permite simular simultáneamente circuitos analógicos y código HDL. Estos simuladores además permiten simular con facilidad pérdidas y componentes parásitos, aumentando la precisión en el modelo de la planta. El mayor inconveniente de estos simuladores es que la velocidad de simulación de estas herramientas es muy lenta. Además, hay pocos simuladores mixtos en el mercado. En el momento de la escritura de este capítulo (Otoño 2011), se han encontrado los simuladores SystemVision [26] y Questa Advanced Simulator [27], ambos de Mentor Graphics. Sin embargo, no se han encontrado simuladores de uso gratuito.

La alternativa es **modelar el convertidor de potencia en HDL**, haciendo que todo el sistema esté descrito en HDL, es decir, realizar un modelo del convertidor de potencia que se comporte como el convertidor real. El regulador ya está descrito de forma nativa en HDL sintetizable, mientras que el modelo del convertidor puede

ser descrito en HDL no sintetizable. Dependiendo del modelado del convertidor en HDL, existen diferentes posibilidades, que se detallan a continuación. Aunque la mayor parte de los contenidos de este capítulo es aplicable para otros lenguajes de descripción de *hardware*, como por ejemplo Verilog, a partir de ahora el capítulo se centrará en el lenguaje VHDL.

- La planta puede ser modelada usando el tipo de señal ***real***, el cual es un tipo numérico con representación en coma flotante soportado por la mayoría de los **simuladores**. Su uso es sencillo, pero su mayor inconveniente es que no puede ser sintetizado.
- El convertidor también puede ser modelado usando el tipo de señal ***float***, el cual es un tipo de datos en coma flotante. Este tipo de datos está descrito en el paquete VHDL2008 *float_pkg* [25]. La ventaja de este paquete es que soporta la **síntesis** de señales de coma flotante siguiendo el estándar IEEE 754 y permitiendo, por tanto, la emulación del sistema. Es importante destacar que la síntesis de este paquete solamente está soportada por algunas herramientas de síntesis, aunque cabe esperar que la compatibilidad aumente progresivamente. Este modelo en coma flotante, al igual que el anterior no sintetizable, permite modelar el convertidor de potencia de forma fácil, siendo su principal ventaja.
- Por último, otra posibilidad es diseñar el convertidor usando **coma fija**. El uso de este tipo de notación implica un esfuerzo de diseño mayor en el modelo. Sin embargo, esta notación permite la emulación del sistema consiguiendo una **velocidad mucho mayor** que la que se consigue usando coma flotante, y **usando muchos menos recursos *hardware***. El mayor esfuerzo en el diseño es debido a que este tipo de notación requiere que el diseñador tenga en cuenta el formato de cada señal, evitando problemas de resolución, desbordamientos, etc. Sin embargo, es importante destacar que el modelo del convertidor normalmente sólo se diseñará una vez, mientras que el regulador será modificado regularmente en la etapa de pruebas.

Como se ha comentado, tanto el modelo *float* como el de coma fija permiten ser emulados. La emulación consiste en implementar el modelo del convertidor en *hardware* real, además del regulador. Normalmente, en emulación el modelo es implementado en una FPGA, dado que es una plataforma óptima para el prototipado rápido. En el apartado 2.3.1 se detallará el modelo de un convertidor de potencia elevador siguiendo las posibilidades descritas.

2.2.1. Ejemplo de aplicación

En este apartado se muestra el convertidor y la aplicación que se van a utilizar como ejemplo a lo largo del capítulo. En particular, se mostrará el proceso de simulación de un **convertidor elevador** (o *boost* en inglés) aplicando **corrección de factor de potencia** o PFC (del inglés *Power Factor Correction*). Se ha escogido la simulación en PFC porque el control sobre la planta incluye dos lazos de dinámicas y frecuencias muy distintas. De esa forma, PFC requiere simulaciones largas con millones de ciclos de reloj. Además, se ha escogido un convertidor elevador ya que es el más habitual en corrección de factor de potencia.

Las características principales de la corrección de factor de potencia están explicadas en la sección 3.1. En corrección de factor de potencia tradicionalmente existen dos lazos. El primero de ellos controla la tensión media de salida, generando un comando de potencia, el cual se utilizará en el segundo lazo. El segundo lazo controla la corriente de entrada para emular cargas resistivas desde el punto de vista de la red, y obtiene como entradas el comando de potencia, y una referencia de corriente a seguir. Los dos lazos descritos tienen **dinámicas muy distintas**, ya que el lazo de corriente es el encargado de conseguir proporcionalidad entre la corriente y tensión de entrada cada ciclo de conmutación (a frecuencias de alrededor de 100 kHz), mientras que el lazo de tensión se encarga de conseguir el balance entre las potencias de entrada y salida a lo largo de varios semiciclos de red, que llegan a 100 Hz.

Los reguladores para PFC permiten rectificar la tensión alterna de entrada controlando simultáneamente la tensión de salida (v_{out}) y la corriente de entrada (i_{in}). El control de la tensión de salida es necesario ya que la carga que se conecta a la salida del convertidor debe recibir tensión continua. Además, se controla que la corriente de entrada sea proporcional a la tensión de entrada (v_g), de forma que se reduzcan los armónicos. Por tanto, hay dos lazos implicados en el controlador: un **lazo de corriente** y un **lazo de tensión**. El primer lazo compara la corriente de entrada con una referencia, la cual se obtiene de la multiplicación de la tensión de entrada y la conductancia de entrada equivalente (g_{in}). Este lazo tiene como salida el ciclo de trabajo del PWM que se conectará al interruptor del *boost*, que normalmente es un transistor MOSFET. Por su parte, el lazo de tensión compara la tensión de salida con una referencia de tensión, que normalmente es constante. Este lazo tiene como salida la conductancia de entrada equivalente, que es usada en el lazo de corriente.

La configuración del convertidor elevador que se va usar en este capítulo está definida en la tabla 2.1. Las funciones de transferencia de las dos plantas que los lazos deben controlar han sido descritas en la literatura, por ejemplo en [28]. En

particular, la planta a controlar en el lazo de corriente que marca la relación entre el ciclo de trabajo y la corriente de entrada está definida en la ecuación (2.1):

$$G_{ID}(s) = \frac{V_{out}}{Ls} \tag{2.1}$$

donde $G_{ID}(s)$ es la función de transferencia de la planta del lazo de corriente una vez realizada la transformada de Laplace, V_{out} es la tensión de salida, y L es la inductancia de la bobina del convertidor.

Por su parte, la planta a controlar en el lazo de tensión que marca la relación entre la conductancia de entrada y la tensión de salida está definida en la ecuación (2.2):

$$G_{VG}(s) = \frac{\frac{v_g^2 R}{2 \cdot V_{out}}}{\frac{RC}{2}s + 1} \tag{2.2}$$

donde $G_{VG}(s)$ es la función de transferencia del lazo de tensión una vez realizada la transformada de Laplace, v_g y V_{out} son las tensiones de entrada y salida del convertidor, R es la resistencia equivalente de la carga en la salida del convertidor, y C es la capacidad del condensador de salida.

Tabla 2.1: Parámetros del convertidor boost.

Parámetro	Valor
f_{sw}	100 kHz
$resolución_{PWM}$	1000 valores
L	5 mH
C	100 μF
P_{out}	300 W
V_{out}	400 V

Se han diseñado unos reguladores para probar el convertidor de potencia. La explicación de estos reguladores no es el objetivo de este capítulo, pero se detallan debido a que sus tiempos de estabilización condicionan el tiempo de simulación necesario para comprobar el correcto funcionamiento de los mismos. La tabla 2.2 muestra los **reguladores digitales** seleccionados, los cuales son reguladores sencillos PID, así como la frecuencia de reloj escogida para la FPGA, que es de 100 MHz. Para diseñar los reguladores digitales, previamente las plantas han sido discretizadas (el periodo de muestreo está detallado en la tabla 2.2), para transformarlas del dominio continuo

Tabla 2.2: Controladores diseñados para el convertidor PFC.

Controlador	Función de transferencia	Periodo de muestreo	Ancho de banda	Tiempo de estabilización
Corriente	$\frac{0,5z-0,4844}{z-1}$	10 μs	6,33 kHz	472 μs
Tensión	$\frac{3,052 \cdot 10^{-5} z - 1,526 \cdot 10^{-5}}{z-1}$	10 ms	6,71 Hz	109 ms
FPGA $f_{CLK} = 100\ MHz$				

al dominio discreto. Una vez discretizadas las plantas, se han diseñado los reguladores usando el lugar de las raíces, usando el método descrito en [6]. Los reguladores PID que han sido escogidos son reguladores conservadores para que su estabilidad sea mayor. Aunque el diseño de reguladores digitales permite realizar técnicas más complejas de control, en este capítulo se ha elegido un controlador sencillo para poder centrarse en su depuración. Por tanto, el objetivo no es diseñar un regulador con ventajas frente a los clásicos, sino ver cómo depurar cualquier regulador implementado en HDL.

Ambos lazos **deben ser simulados antes de ser probados en el convertidor real**. Sin embargo, la simulación requerida es significativamente larga. La simulación debe realizar los cálculos de un lazo de alta frecuencia, el cual controla la dinámica de la corriente de entrada. Sin embargo, la simulación también debe ser lo suficientemente larga para poder observar la evolución de la tensión de salida a lo largo del tiempo. En el sistema propuesto, el periodo de reloj de la FPGA es de 10 ns, mientras que el tiempo de estabilización del lazo de corriente es de 472 μs y 109 ms para el lazo de tensión. Por esta razón, y debido a la diferencia en varios órdenes de magnitud en dichos parámetros, las simulaciones deben calcular cientos de milisegundos o incluso algunos segundos, lo que corresponde a **decenas de millones de ciclos de reloj**. La razón principal para aplicar técnicas de simulación rápida o emulación es la **aceleración** que se puede obtener en las etapas de prueba de los reguladores.

2.3. Implementación de los modelos de la planta

En este apartado se mostrará cómo modelar un convertidor elevador, así como los detalles de su implementación, usando las diferentes posibilidades de simulación descritas en el apartado 2.2.

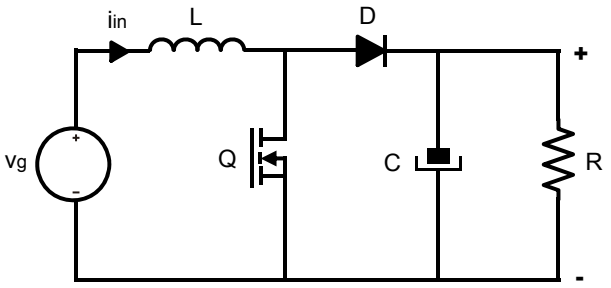


Figura 2.4: Topología de un convertidor elevador

2.3.1. Modelo de un convertidor boost

La aplicación, como se ha comentado en el apartado anterior, es la rectificación de la red eléctrica usando técnicas de PFC y un convertidor elevador. Para realizar una simulación íntegra en VHDL, se debe hacer un modelo de la planta, es decir, de dicho convertidor elevador, ver figura 2.4.

El modelo de la planta debe calcular en cada instante de tiempo la **tensión de salida** v_{out} y la **corriente de entrada** i_{in}, que es igual a la corriente que circula por la bobina i_L. La tensión de la bobina de entrada está definida por la ecuación (2.3):

$$v_L = L \cdot \frac{di_L}{dt} \tag{2.3}$$

La ecuación (2.3) puede transformarse en una ecuación en diferencias en la cual la corriente de entrada en el tiempo k es definida como (2.4):

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot v_L \tag{2.4}$$

$i_L(k)$ es la corriente de entrada en el instante de tiempo k, Δt es el paso de integración en el cálculo de las variables de estado, y v_L es la tensión de la bobina.

De la misma forma, la corriente que pasa a través del condensador de salida (i_C) está definida como (2.5):

$$i_C = C \cdot \frac{dv_{out}}{dt} \tag{2.5}$$

y transformándola a ecuaciones en diferencias, la tensión de salida en el tiempo k está definida en (2.6):

$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot i_C \tag{2.6}$$

Para facilitar la síntesis del modelo propuesto se ha utilizado un tiempo de integración (Δt) fijo. Por tanto, $\frac{\Delta t}{L}$ y $\frac{\Delta t}{C}$ son constantes. Por otra parte, i_C es la corriente en el condensador, la cual está determinada por la carga que se conecte a la salida del convertidor. En particular, i_C es igual a la corriente en la carga (i_R) cuando el transistor está cerrado, mientras que será $i_L - i_R$ si el transistor está abierto. La expresión $i_R = \frac{v_{out}}{R}$ puede ser utilizada si la carga conectada a la salida es resistiva. Sin embargo, en el modelo propuesto se deja i_R como variable independiente, permitiendo modelar cualquier tipo de carga.

Cuando el transistor está abierto, la corriente de entrada (i_L) puede ser positiva de forma que el diodo conduzca (CCM o en inglés *Continuous Current Mode*), o puede ser igual a cero, de forma que el diodo no conduzca (DCM, o en inglés *Discontinuous Current Mode*). Debido a ello, hay tres posibilidades en el cálculo de las variables de estado: **transistor cerrado**, **transistor abierto en CCM** o **transistor abierto en DCM**, las cuales están descritas en las ecuaciones (2.7), (2.8) y (2.9):

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot v_g \\ v_{out}(k) &= v_{out}(k-1) - \frac{\Delta t}{C} \cdot i_R \end{aligned} \tag{2.7}$$

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (v_g - v_{out}) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{2.8}$$

$$\begin{aligned} i_L(k) &= 0 \\ v_{out}(k) &= v_{out}(k-1) - \frac{\Delta t}{C} \cdot i_R \end{aligned} \tag{2.9}$$

Solamente uno de estos tres conjuntos de ecuaciones debe ser calculado en cada ciclo de reloj, por lo que se deben realizar dos multiplicaciones anidadas cada ciclo, aparte de sumas y restas.

Las ecuaciones en diferencias (2.7), (2.8) y (2.9) pueden ser utilizadas en los tres modelos VHDL que se comentaron en el apartado 2.2: *real*, *float* y coma fija. El uso de *real* o *float* está motivado por la simplicidad del diseño, así que es razonable utilizar las ecuaciones descritas, sin realizar optimizaciones que compliquen el diseño. Sin embargo, el uso de coma fija requiere que el diseñador seleccione el formato de cada señal, lo que implica un diseño más complejo pero mucho más rápido en emulación. Siguiendo la filosofía de optimizar la velocidad del modelo en coma fija, se pueden usar algunas transformaciones.

En vez de calcular i_L, se puede calcular i_L^* aplicando la transformación descrita en la ecuación (2.10):

$$i_L^* = \frac{L}{\Delta t} \cdot i_L \tag{2.10}$$

La ventaja de usar esta transformación es que evitamos realizar la multiplicación de la ecuación 2.4. Por tanto, el cálculo de la corriente de entrada del convertidor elevador queda definida como (2.11):

$$i_L^*(k) = i_L^*(k-1) + v_L \tag{2.11}$$

Se puede realizar la misma transformación con la tensión de salida (2.12):

$$v_{out}^* = \frac{C}{\Delta t} \cdot v_{out} \tag{2.12}$$

La tensión de salida del convertidor queda definida entonces como (2.13):

$$v_{out}^*(k) = v_{out}^*(k-1) + i_C \tag{2.13}$$

El cálculo de la tensión de salida depende de la corriente que cruza el condensador (i_C). Se debe realizar otra transformación si se desea utilizar en la ecuación i_C^* en vez de i_C. Para ello, v_{out}^* debe transformarse en v_{out}^{**} según (2.14):

$$v_{out}^{**} = \frac{L}{\Delta t} v_{out}^* = \frac{C}{\Delta t}\frac{L}{\Delta t} \cdot v_{out} \tag{2.14}$$

Aplicando la anterior transformación, el cálculo de la tensión de salida queda expresada en (2.15):

$$v_{out}^{**}(k) = v_{out}^{**}(k-1) + i_C^* \tag{2.15}$$

Las ecuaciones (2.11) y (2.15) no usan multiplicaciones, y por tanto la frecuencia máxima de funcionamiento es significativamente mayor, además de usar menos recursos *hardware*. Al igual que en el modelo básico, i_C^* y v_L dependen del estado del transistor y del modo de conducción (CCM o DCM). Por tanto, las ecuaciones que se deben implementar en *hardware* son:

$$\begin{aligned} i_L^*(k) &= i_L^*(k-1) + v_g \\ v_{out}^{**}(k) &= v_{out}^{**}(k-1) - i_R^* \end{aligned} \tag{2.16}$$

$$\begin{aligned} i_L^*(k) &= i_L^*(k-1) + v_g - v_{out} \\ v_{out}^{**}(k) &= v_{out}^{**}(k-1) + i_L^* - i_R^* \end{aligned} \tag{2.17}$$

$$\begin{aligned} i_L^*(k) &= 0 \\ v_{out}^{**}(k) &= v_{out}^{**}(k-1) - i_R^* \end{aligned} \tag{2.18}$$

Como se puede observar en la ecuación (2.17), i_L^* depende de v_{out}, por lo que la transformación expresada en (2.14) debe ser restaurada para poder conocer el valor

de v_{out} a partir de v_{out}^{**}. Esta restauración conlleva una multiplicación que deberá realizarse en cada ciclo de reloj (2.19):

$$v_{out} = \frac{\Delta t}{C}\frac{\Delta t}{L} \cdot v_{out}^{**} \tag{2.19}$$

No es necesario restaurar la transformación realizada en la corriente (2.10), ya que las ecuaciones (2.16), (2.17) y (2.18) no usan directamente la corriente i_L. De este modo, solamente es necesario restaurar la transformación de la tensión, **evitando realizar una de las dos multiplicaciones originales**. Esta simplificación en *hardware* consigue reducir el tiempo de cálculo en el lazo, por lo que la frecuencia máxima del lazo aumenta.

Aunque no es necesario el cálculo de i_L para completar el lazo, puede ser útil saber su valor sin ninguna escala, es decir, en amperios. Para ello habría que deshacer la ecuación (2.10) añadiendo una segunda multiplicación. Sin embargo, esta multiplicación no estaría en el camino crítico y no afectaría a la frecuencia máxima de funcionamiento.

La figura 2.9 muestra la implementación de las ecuaciones discretas, y las optimizaciones descritas. El uso de estas transformaciones para acelerar el proceso de emulación en *hardware* es una **aportación original** de esta tesis.

2.3.2. Modelo mixto analógico-digital

La simulación mixta analógica digital permite al diseñador realizar con relativa facilidad un sistema de pruebas. Este tipo de simuladores normalmente permiten dibujar un circuito, mediante el método de **«arrastrar y soltar»**, insertando componentes tales como condensadores, bobinas, resistencias y ADCs. Estos componentes suelen estar descritos en VHDL-AMS, que es una extensión al lenguaje VHDL para describir componentes analógicos y mixtos (analógico-digitales). Sin embargo, la implementación de estos componentes es casi siempre transparente para el diseñador, sólo siendo necesario modificarla en algunos casos. El regulador, que se ha diseñado en VHDL, se puede añadir como un componente gráfico más, permitiendo dibujar todo el circuito (ver figura 2.5). La implementación y simulación mediante este tipo de herramientas no es compleja. Sin embargo, el tiempo de simulación es notablemente largo. Se ha elegido la herramienta SystemVision 5.7 de Mentor Graphics para implementar el sistema de simulación mixta y realizar pruebas.

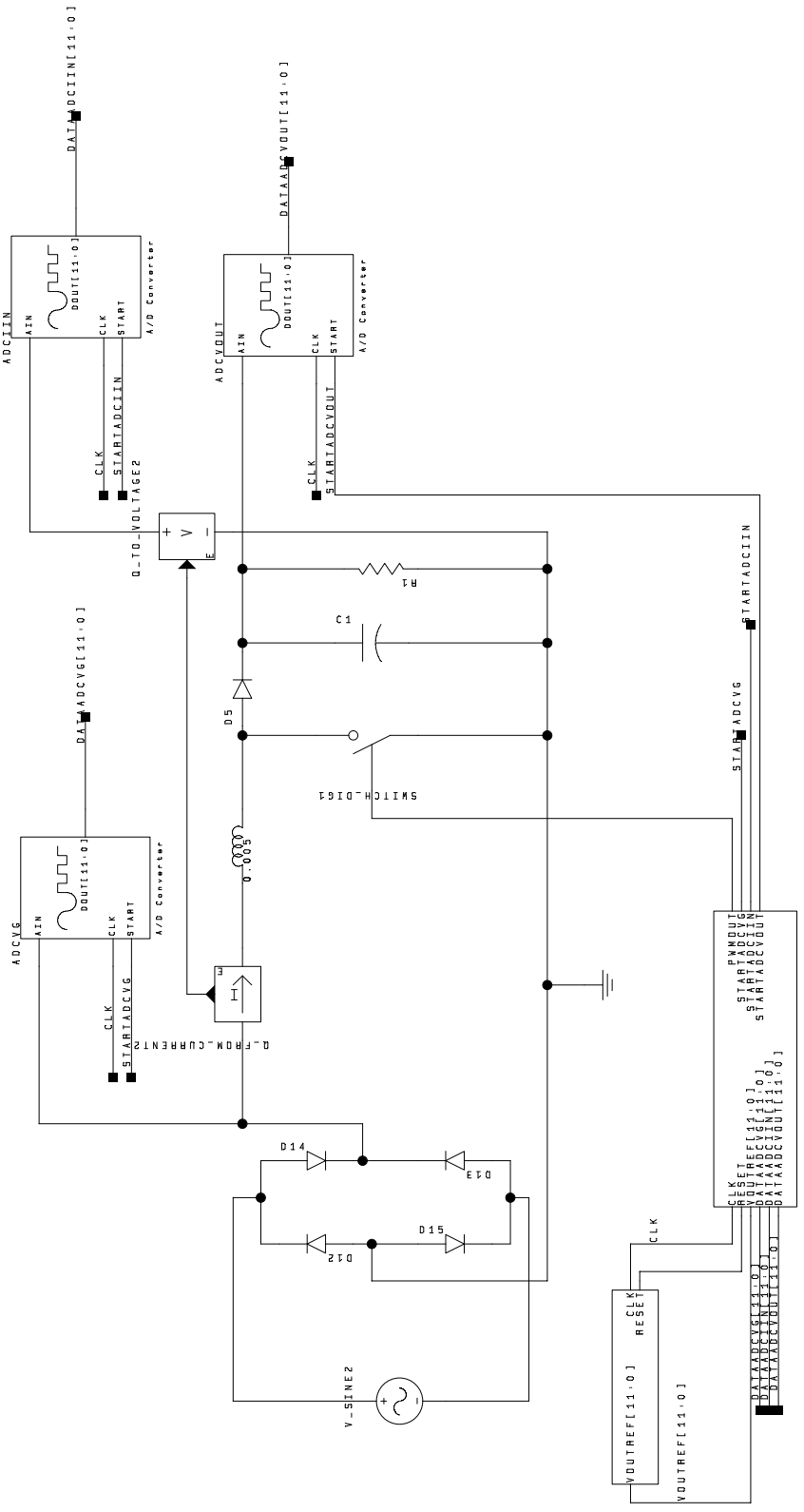


Figura 2.5: Esquemático del sistema implementado con SystemVision.

2.3.3. Modelo *real*

El modelo del convertidor elevador con señales de tipo *real* se ha implementado usando las ecuaciones (2.7), (2.8) y (2.9). El modelo también es sencillo, usando solamente cuatro multiplexores y dos multiplicadores, además de varios sumadores y registros.

El código 2.1 muestra el proceso *DIFFEQ*, el cual se encarga de actualizar el valor de la corriente de entrada y tensión de salida en cada ciclo de reloj, según los valores *iLAdd* y *voutAuxAdd* y en donde *dtL* y *dtC* son las constantes $\frac{\Delta t}{L}$ y $\frac{\Delta t}{C}$ respectivamente.

```
DIFFEQ: process(Clk, Reset)
-- Update of Vout and Iin each clock cycle
begin
        if Reset = '1' then
                voutAux <= VOINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then
                iL <= iL + iLAdd*dtL;
                voutAux <= voutAux + voutAuxAdd*dtC;
        end if;
end process DIFFEQ;
```

Código 2.1: Modelo *real* del convertidor elevador. Actualización de señales

Los valores *iLAdd* y *voutAuxAdd* dependen del estado del transistor y el modo de conducción según muestra el código 2.2. El modelo completo del convertidor *boost* puede encontrarse en el Anexo A.1.1.

```
SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
-- Selection (multiplexer) of values to be added to input current and
-- output voltage
begin
        if Mosfet = '1' then -- Closed switch
                iLAdd <= Vg;
                voutAuxAdd <= -(Ir);
        else -- Open switch
                if iL > 0.0 then -- CCM
                        iLAdd <= (Vg - voutAux);
                        voutAuxAdd <= (iL - Ir);
                else -- DCM
                        iLAdd <= 0.0;
                        voutAuxAdd <= -(Ir);
                end if;
        end if;
end process SWITCHMUX;
```

Código 2.2: Modelo *real* del convertidor elevador. Multiplexor

Aunque el modelo con tipo *real* no se puede sintetizar, la división en dos procesos se ha realizado así para seguir la misma estructura de los modelos sintetizables que, como se comentará, optimizan el área ocupada en *hardware*.

2.3.4. Modelo *float*

El modelo con tipo *float* puede ser simulado pero también sintetizado. Este modelo es prácticamente idéntico al que usa señales *real*, ya que tanto las señales *real* como las señales *float* son de coma flotante. Sin embargo, éste usa el paquete *float_ pkg* de la biblioteca *VHDL-2008 Support Library* [25]. Este paquete define los tipos sintetizables *float32* y *float64*, los cuales son señales de 32 y 64 bits respectivamente, siguiendo el estándar IEEE 754, de coma flotante. El mayor problema de esta biblioteca es que, por ahora, **pocos simuladores y sintetizadores la soportan**. Uno de los sintetizadores que pueden manejarla es Synplify Premier de Synopsys, el cual se ha utilizado para hacer pruebas. Aunque el paquete define señales de 32 y 64 bits, solamente se han utilizado señales de 32 bits para reducir el *hardware* necesario el cual, como se comentará en los resultados de implementación, es notablemente grande incluso usando coma flotante de 32 bits. También es posible determinar anchos personalizados para las señales, no siendo necesario ajustarse a los tamaños estándar. Sin embargo, la principal motivación para usar el paquete *float_ pkg* es la inmediatez en el diseño, y hacer un estudio del ancho óptimo de las señales es contrario a dicha motivación. Al igual que en el modelo *real*, el código con *float* se divide en dos procesos, mostrados en los códigos 2.3 y 2.4. El modelo completo se encuentra en el anexo A.1.2.

```
DIFFEQ: process(Clk, Reset)
-- Update of Vout and Iin each clock cycle
begin
        if Reset = '1' then
                voutAux <= VOINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then
                iL <= iL + iLAdd * dtL;
                voutAux <= voutAux + voutAuxAdd * dtC;
        end if;
end process DIFFEQ;
```

Código 2.3: Modelo *float* del convertidor elevador. Actualización de señales

Vg es la tensión de entrada, que en la mayoría de los casos será la de la red eléctrica una vez pasada por un puente de diodos. Para simplificar la generación de la sinusoide v_g en emulación, ésta se almacena en 1 000 pasos diferentes en una BRAM (*Block RAM*) de la FPGA, por lo que en cada ciclo de cálculo, un nuevo valor de v_g es cargado desde la memoria.

La división en dos procesos, uno que selecciona el valor a sumar y otro que realiza la suma, se ha realizado para optimizar el área del diseño. Otra posibilidad sería implementar el modelo directamente mediante las ecuaciones (2.7), (2.8) y (2.9), lo cual origina el código 2.5.

```
SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
-- Selection (multiplexer) of values to be added to input current and
-- output voltage
begin
        if Mosfet = '1' then -- Closed switch
                iLAdd <= Vg;
                voutAuxAdd <= -(Ir);
        else -- Open switch
                if gt(iL, CZERO) then -- CCM (gt: greater than)
                        iLAdd <= (Vg - voutAux);
                        voutAuxAdd <= (iL - Ir);
                else -- DCM
                        iLAdd <= CZERO;
                        voutAuxAdd <= -(Ir);
                end if;
        end if;
end process SWITCHMUX;
```

Código 2.4: Modelo *float* del convertidor elevador. Multiplexor

```
DIFFANDMUX: process(Clk, Reset)
begin
        if Reset = '1' then
                voutAux <= VOINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then
                if Mosfet = '1' then -- Closed switch
                        iL <= iL + Vg * dtL;
                        voutAux <= voutAux - Ir * dtC;
                else -- Open switch
                        if gt(iL, CZERO) then -- CCM (gt: greater than)
                                iL <= iL + (Vg - voutAux) * dtL;
                                voutAux <= voutAux + (iL - Ir) * dtC;
                        else -- DCM
                                iL <= iL + CZERO; -- iL <= CZERO
                                voutAux <= voutAux -(Ir) * dtC;
                        end if;
                end if;
        end if;
end process DIFFANDMUX;
```

Código 2.5: Modelo *float* no optimizado del convertidor elevador.

En la figura 2.6 se muestra el *hardware* que se implementaría con el código 2.5 para calcular la corriente de entrada. Esta implementación utiliza dos multiplexores, tres sumadores y dos multiplicadores. Sin embargo, usando la optimización comentada (códigos 2.3 y 2.4), que consiste en el cambio de orden entre los multiplexores y los multiplicadores y sumadores, se reduce el *hardware* necesario, ya que se eliminan un multiplicador y un sumador, figura 2.7. Aunque las figuras 2.6 y 2.7 solamente muestran los circuitos necesarios para calcular la corriente de entrada, la misma optimización se realiza en el cálculo de la tensión de salida.

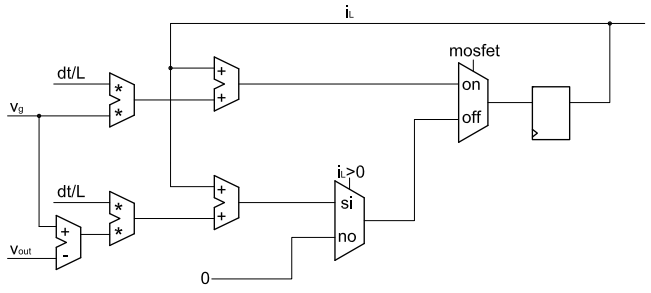


Figura 2.6: Implementación directa de las ecuaciones (2.7), (2.8) y (2.9).

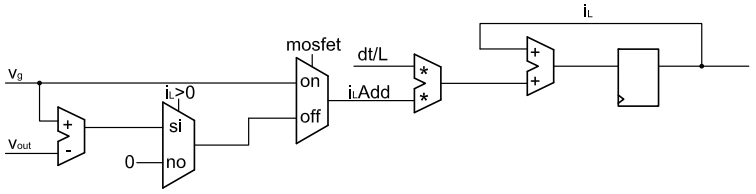


Figura 2.7: Implementación optimizada de las ecuaciones (2.7), (2.8) y (2.9).

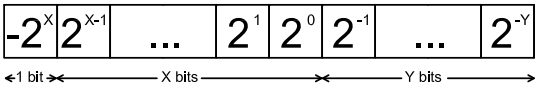


Figura 2.8: Formato de una señal en QX.Y

2.3.5. Modelo en coma fija

El modelo en coma fija requiere **mayor tiempo de diseño pero permite conseguir los mejores resultados en área y frecuencia**. Este modelo usa las ecuaciones (2.16), (2.17) y (2.18). Las señales en coma fija pueden seguir la notación QX.Y. Una señal QX.Y tiene X bits de parte entera e Y bits de parte decimal, aparte de 1 bit adicional para el signo, siguiendo la notación de complemento a dos, como se puede ver en la figura 2.8. Es decir, una señal de tipo Q5.3 tiene $1 + 5 + 3$ bits.

La tabla 2.3 muestra el formato y escala de las señales que se han utilizado en el modelo en coma fija. Para traducir el valor literal de una señal QX.Y en su valor real, dicho valor debe ser multiplicado por la escala de la señal y por 2^{-Y}. Algunas señales, como v_{out} no tienen escala, así que su valor solamente debe ser multiplicado por 2^{-Y} para obtener su valor en voltios o amperios. Por ejemplo, el valor "0100000000001" en la señal v_{out} representa 256,125 V. Sin embargo, la señal i_R^* tiene los mismos bits, pero además tiene escala, ya que se ha utilizado en ella la transformación (2.10). La escala es una constante por la que debe ser multiplicado el valor de la señal para obtener su valor real en voltios o amperios. En el caso de i_R, su escala viene determinada por Δt, que es el tiempo de integración del modelo (periodo del reloj de la FPGA) y L, que es la inductancia de la bobina. Por tanto, el mismo valor de "0100000000001" en la

Tabla 2.3: Formato de las señales del modelo en coma fija.

Señal	Número de bits	Formato	Escala	Rango equivalente (3 decimales)	Resolución
v_g	13	9.3	-	$\pm$ 511,875 V	0,125 V
v_{out}	13	9.3	-	$\pm$ 511,875 A	0,125 V
i_R^*	13	22.-10	$\frac{\Delta t}{L}$	$\pm$ 8,387 A	$2{,}048 \cdot 10^{-3}$ A
v_{out}^*	34	43.-10	$\frac{\Delta t}{L}\frac{\Delta t}{C}$	$\pm$ 1 759,219 V	$2{,}048 \cdot 10^{-7}$ V
$v_{out}Sat^*$	18	43.-26	$\frac{\Delta t}{L}\frac{\Delta t}{C}$	$\pm$ 1 759,205 V	0,013 V
i_L^*	26	22.3	$\frac{\Delta t}{L}$	$\pm$ 8,389 A	$2{,}5 \cdot 10^{-7}$ A
i_LSat^*	18	22.-5	$\frac{\Delta t}{L}$	$\pm$ 8,389 A	$6{,}4 \cdot 10^{-5}$ A

señal i_R representa 4,196352 A. En la tabla 2.3 también se han añadido la resolución y rango de representación de cada señal que usa el modelo en coma fija.

La figura 2.9 muestra el esquemático del modelo del convertidor elevador en coma fija. La parte izquierda de la figura muestra el *hardware* necesario para calcular la corriente de entrada. i_LAdd^* es la cantidad que debe añadirse al anterior valor de i_L^*. Este incremento depende del estado de dos multiplexores, los cuales comprueban el estado del transistor y el modo de conducción (CCM o DCM). El cálculo de la corriente también tiene en cuenta la tensión de entrada, la cual viene dada normalmente por un ADC. Se ha escogido un tamaño de entrada de 12 bits más uno de signo porque la mayoría de los ADCs de bajo coste ofrecen su salida en 12 bits sin signo. Además, se añade 1 bit de signo (siempre positivo) para convertirlo a formato QX.Y.

La cantidad a añadir a la corriente (i_LAdd^*) está expresada en voltios, por lo que se usa la transformación (2.10) para poder ser sumada directamente. El cálculo de la corriente i_L^* se guarda en 26 bits, obteniendo una resolución de $2{,}5 \cdot 10^{-7}A$. Esta corriente almacenada y la corriente de la carga i_R son las entradas del *hardware* que calcula la tensión de salida del sistema (parte derecha de la figura). Sin embargo, no se usan los 26 bits de i_L^* como entrada a la siguiente etapa, sino que se usan los 13 bits más significativos. Este truncado se utiliza para reducir el número de bits implicado en los cálculos de la segunda parte (cálculo de la tensión de salida). Es importante destacar que la reducción de 26 a 13 bits no reduce notablemente la precisión del sistema. La corriente i_L^* expresada en 13 bits tiene una resolución de 2,048 mA por lo que permite hacer cálculos precisos para calcular la tensión de salida. Aún así, la corriente de entrada internamente se almacena con una resolución de 0,25 μA usando

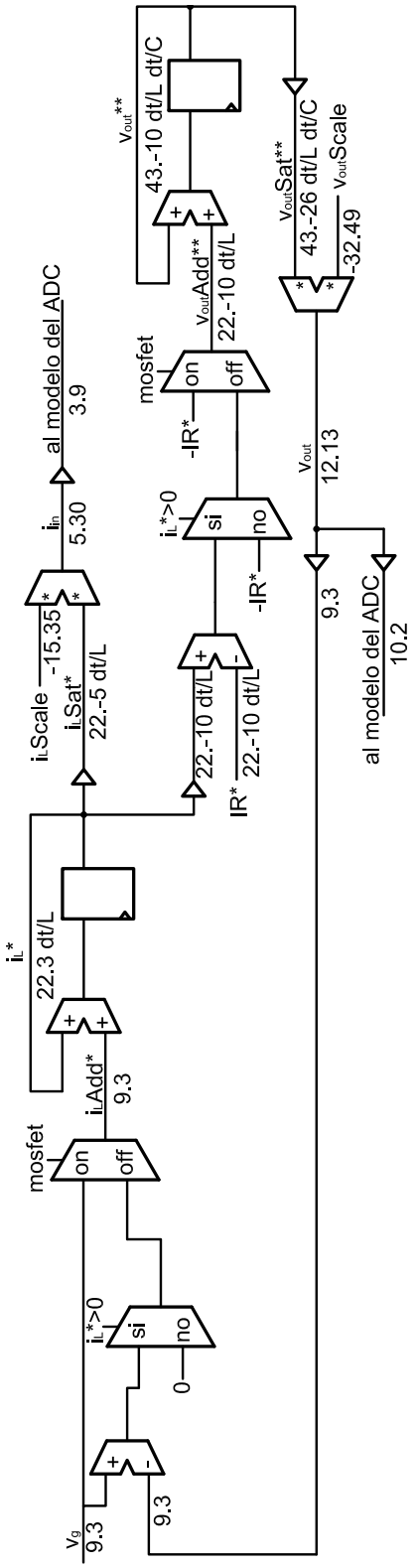


Figura 2.9: Esquemático del circuito implementado en coma fija.

los 26 bits. El cálculo interno en 26 bits permite que la **integración** de la corriente sea mucho **más precisa**. Esto es debido a que si la corriente de entrada cambia en una cantidad menor a la resolución en 13 bits, la salida de la corriente mantendrá su valor, pero sí cambiará el registro de 26 bits. Múltiples cambios pequeños en el tiempo se **acumularán** en el registro de 26 bits y podrán originar un cambio en la salida de la corriente. El problema de resolución en las señales se comenta más extensamente en el aparatado 2.6. Por último, la corriente en 13 bits es restada a i_R^*, la cual es una entrada del modelo que representa la corriente de la carga conectada al convertidor.

Al igual que en el cálculo de la corriente, el valor a añadir a la tensión de salida está controlado mediante dos multiplexores que dependen del estado del transistor y del modo de conducción. El incremento $v_{out}Add^{**}$ es una señal de corriente en la escala de i_L^*, así que para ser sumada de nuevo se debe hacer una transformación, siendo en este caso la transformación (2.14). Por último, el lazo debe ser cerrado debido a que la tensión de salida influye en el cálculo de la corriente del siguiente ciclo de cálculo. Ya que la señal v_{out}^{**} no está expresada en voltios, sino que tiene una doble escala producida por la transformación (2.14), ésta debe ser restaurada con la transformación (2.19). v_{out}^{**} está expresado en 34 bits pero, en vez de multiplicar su valor directamente por su doble escala, sólo se seleccionan los 18 bits más significativos de v_{out}^{**}. La razón de este truncado es que los multiplicadores de la FPGA que se ha usado para las pruebas (Xilinx Spartan 3) tienen **multiplicadores embebidos** de 18x18 bits. Su uso acelera sustancialmente el circuito implementado. Por tanto, los 18 bits más significativos de v_{out}^{**} se multiplican por su escala, también expresada en 18 bits, obteniéndose la tensión de salida en voltios. Por último, la tensión de salida en voltios vuelve a ser truncada a 13 bits para poder ser restada a la tensión de entrada en el siguiente ciclo de cálculo.

En el cálculo de las variables, solamente es necesaria una multiplicación, en vez de dos como en la solución propuesta para *real* y *float*, aumentando la frecuencia máxima de trabajo. En la parte superior derecha del esquemático se puede observar que hay otra multiplicación para convertir i_L^* a amperios y realimentar el controlador PFC. Sin embargo, este segundo multiplicador no está en el camino crítico, por lo que no afecta a la frecuencia máxima de trabajo.

El modelo tiene dos salidas, v_{out} e i_{in}, las cuales se envían al modelo del ADC, que se comentará posteriormente. Además, como se ha visto, el modelo tiene tres entradas: i_R^*, v_g y $mosfet$. i_R^* es la corriente de la carga, la cual se deja como variable independiente para poder modelar cualquier carga de forma variable. $mosfet$ es el estado del transistor controlado por el regulador PFC. Al igual que en el modelo con señales *float*, v_g es precalculada en BRAMs.

El esquemático descrito está codificado en dos procesos, al igual que los modelos *real* y *float*. El código 2.6 muestra los sumadores y registros de la corriente de entrada y tensión de salida. Por su parte, el código 2.7 muestra los cuatro multiplexores que contiene el esquema.

```
DIFFEQ: process(Reset, Clk)
-- Update of Vout and Iin each clock cycle
begin
        if Reset = '1' then
                voutAux <= VOAUXINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then
                iL <= iL + iLAdd;
                voutAux <= voutAux + voutAuxAdd;
        end if;
end process DIFFEQ;
```

Código 2.6: Modelo QXY del convertidor elevador. Actualización de señales

```
SWITCHMUX : process (Mosfet, Vg, Ir, iL, voutFeedback)
begin
        if Mosfet = '1' then -- Closed switch
                iLAdd <= Vg;
                voutAuxAdd <= -Ir;
        else -- Open switch
                if iL > conv_std_logic_vector(0, iL'length) then -- CCM
                        iLAdd <= Vg - voutFeedback;
                        -- iL is truncated to be in the same scale than Ir
                        voutAuxAdd <= iL(25 downto 13) - Ir;
                else -- DCM
                        iLAdd <= (others => '0');
                        voutAuxAdd <= - Ir;
                end if;
        end if;
end process SWITCHMUX;
```

Código 2.7: Modelo QXY del convertidor elevador. Multiplexor

Por último, el código 2.8 muestra las operaciones de saturación y transformaciones de escala necesarias para completar el lazo. El código completo de esta implementación se muestra en el anexo A.1.3.

Como se puede observar, el modelo en coma fija es más complejo que el modelo *real* o *float* debido a que hay que tener en cuenta en cada operación el formato de cada señal. Sin embargo, como se mostrará, su frecuencia máxima de trabajo será mucho mayor.

2.3.6. Modelo en coma fija usando la biblioteca *sfixed*

El modelo en coma fija permite ajustar el tamaño de cada registro, posibilitando llegar a una relación óptima entre precisión y velocidad de simulación. Además, el uso

```
-- Q43.-26 dt/C dt/L. Truncated in order to fit in a 18x18 multiplier
voutAuxSat <= voutAux(33 downto 16);
-- Q12.23 = Q43.-26 * Q-32.49
voutScaled <= voutAuxSat * VOUTSCALE;
-- To be added with Vg. Q9.3. If vout > 512 V, voutFeedback overflows.
voutFeedback <= voutScaled(32 downto 20);
-- Q10.2 without sign bit
Vout <= voutScaled(32 downto 21) when voutScaled(32) = '0' else (others => '0');
-- Q22.-5 dt/L. Truncated in order to fit in a 18x18 multiplier
iLSat <= iL(25 downto 8);
-- Q5.30 = Q22.-5 * Q-18.35
iinScaled <= iLSat * IINSCALE;
-- Q3.9 without sign bit
Iin <= iinScaled(32 downto 21) when iinScaled(32) = '0' else (others => '0');
```

Código 2.8: Modelo QXY del convertidor elevador. Transformaciones

de coma fija permite que el modelo sea sintetizado con gran facilidad. Sin embargo, la implementación en coma fija no es una tarea trivial, necesitando realizar a mano numerosas conversiones de tamaños, concatenaciones, etc. Debido a esta dificultad, este apartado propone el uso de una biblioteca que facilita la implementación de aritmética en coma fija.

La biblioteca que se propone es *fixed* dentro del proyecto *VHDL-2008 Support Library* [25]. En particular se utiliza la biblioteca *sfixed* la cual permite usar aritmética con signo. Es importante destacar que las ventajas de esta propuesta se obtienen únicamente en la etapa de implementación, mientras que la etapa de diseño es completamente similar a la propuesta en el apartado anterior. Por tanto, el cálculo de los tamaños de cada registro, y el esquemático presentado en la figura 2.9 son necesarios usando esta biblioteca.

La actualización de los registros que guardan las variables de estado v_{out} e i_L se muestra en el siguiente código:

```
DIFFEQ: process(Reset, Clk)
-- Update of Vout and Iin each clock cycle
begin
        if Reset = '1' then
                voutAux <= VOAUXINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then
                iL <= resize(iL + iLAdd,iL);
                voutAux <= resize(voutAux + voutAuxAdd,voutAux);
        end if;
end process DIFFEQ;
```

Código 2.9: Modelo *sfixed* del convertidor elevador. Actualización de señales

Como se puede observar, el código es similar al código en coma fija realizado sin ayuda de bibliotecas (Código 2.6). El único cambio presentado es que el resultado de una operación puede adaptarse al tamaño deseado usando la función *resize*. El

primer parámetro de esta función contiene el valor a convertir, mientras que el segundo obtiene la señal cuyo formato será tomado como referencia en el resultado. Por tanto, el segundo parámetro sólo será usado para determinar el tamaño que tendrá el resultado de la conversión. Obviamente el tamaño pedido debe ser suficiente para almacenar el valor deseado, por lo que el cálculo de tamaños de los registros debe realizarse.

Al igual que en apartados anteriores, las señales iLAdd y voutAuxAdd son calculadas aparte, en el siguiente código:

```
SWITCHMUX : process (Mosfet, vVg, vIr, iL, voutFeedback)
        begin
                if Mosfet = '1' then -- Closed switch
                        iLAdd <= resize(vVg, iLAdd);
                        voutAuxAdd <= resize(- vIr, voutAuxAdd);
                else -- Open switch
                        if iL > 0 then -- CCM
                                iLAdd <= resize(vVg - voutFeedback,iLAdd);
                                voutAuxAdd <= resize ( resize(iL, voutAuxAdd)
                                        - vIr, voutAuxAdd);
                        else -- DCM
                                iLAdd <= to_sfixed(0.0, iLAdd);
                                voutAuxAdd <= resize(-vIr, voutAuxAdd);
                        end if;
                end if;
        end process SWITCHMUX;
```

Código 2.10: Modelo QXY del convertidor elevador. Multiplexor

Como se puede observar, los códigos 2.9 y 2.10 de *sfixed* son similares a los códigos 2.6 y 2.7 de QX.Y. El mayor cambio reside en el uso de operaciones aritméticas de diferentes anchos. En el modelo en QX.Y para operar con dos señales no alineadas a la derecha (por ejemplo iL y voutAuxAdd), es necesario escoger los bits necesarios de cada señal para que estén alineadas. Sin embargo, en el modelo *sfixed* se utiliza la función *resize*, la cual permite cambiar el tamaño de una señal automáticamente. A priori no es una gran ventaja, pero facilita futuros cambios. Por ejemplo, si se desea cambiar el tamaño de una señal a posteriori, la resta de iL y voutAuxAdd seguirá siendo válida ya que la función *resize* se adapta automáticamente a los nuevos tamaños de señales. En cambio, en el modelo QX.Y, sería necesario alinear nuevamente las dos señales dependiendo del tamaño actual de cada señal.

La última ventaja reseñable de la biblioteca *sfixed* es la posibilidad de realizar automáticamente el redondeo de los resultados de operaciones aritméticas. A la hora de declarar la biblioteca se puede definir si los resultados de las operaciones de reescalado (*resize*) deben ser redondeados o no. El redondeo aumenta ligeramente la precisión del modelo. En contrapartida, el redondeo aumenta ligeramente el *hardware* necesario para emulación, aumenta el tiempo de simulación y reduce frecuencia máxima de

implementación. En el caso de QX.Y, esta tarea debería hacerse a mano utilizando un multiplexor y un sumador dependiente del bit más significativo no usado en el resultado.

2.3.7. Modelo del ADC

El controlador necesario para PFC descrito en la figura 3.1 necesita tres conversores analógico-digitales para medir las tensiones de entrada y salida, y la corriente de entrada en el convertidor. El controlador, por tanto, lleva implementada la interfaz de comunicación con los conversores. Esta interfaz también debe probarse en la simulación.

La inserción de un modelo de ADC es importante por dos motivos: **convertir la salida del modelo del convertidor** al formato esperado por el controlador, y **modelar retrasos** típicos de un ADC.

El sistema de simulación mixta diseñado con SystemVision debe convertir las señales declaradas como analógicas que representan las tensiones y corrientes en valores digitales. Además, los modelos realizados en VHDL también necesitan un modelo del ADC. En particular, los convertidores descritos con señales *real* y *float* deben añadir un modelo de ADC para convertir sus salidas a coma fija, que será la notación que seguramente use el controlador. Por su parte, el modelo del convertidor en coma fija también debe usar un modelo del ADC. En este caso, el ADC modela los retrasos que se originan en un sistema real y, por otra parte, puede que las señales de salida del modelo del convertidor no compartan el mismo formato QX.Y que las señales que espera el controlador.

Si se opta por usar un simulador mixto, es muy probable que la herramienta ofrezca implementaciones de ADCs, por lo que el diseñador solamente tiene que insertar uno y conectarlo al circuito, configurando las tensiones de referencia, ancho de palabra, etc. En cambio, si se elige simulación o emulación en VHDL, el diseñador debe realizar un modelo del ADC. El código 2.11 muestra el modelo del ADC que se debe utilizar para convertir señales de tipo *real* en coma fija.

El modelo del ADC espera un flanco de subida en la señal *Start*. Esto se consigue con dos registros (no mostrados en el código 2.11), y comprobando la secuencia 0->1. Cuando se detecta se toma una muestra de la señal de entrada *AnalogIn*, saturándola entre 0 y $2^{NBITS} - 1$, teniendo en cuenta que la señal digital convertida será de *NBITS*. En ese momento se activa un contador que se incrementará en cada ciclo de

```
prDelay : process(Reset, Clk)
begin
        if Reset = '1' then
                counter <= 0;
                sample <= 0.0;
                DataOut <= (others => '0');
        elsif rising_edge(Clk) then
                if StartR = '1' and startR2 = '0' then
                        counter <= 1;
                        if (AnalogIn < 0.0) then
                                sample <= 0.0;
                        elsif (AnalogIn >= real(2**NBITS-1)) then
                                sample <= real(2**NBITS-1);
                        else
                                sample <= AnalogIn;
                        end if;
                elsif counter = CYCLESDELAY then
                        DataOut <= conv_std_logic_vector(floor(sample), NBITS);
                elsif counter /= 0 then
                        counter <= counter + 1;
                end if;
        end if;
end process prDelay;
```

Código 2.11: Modelo de un ADC (conversión de *real* a QX.Y)

reloj. Cuando el contador llegue a un retraso predefinido, el modelo del ADC muestra el valor convertido a la salida: *DataOut*.

El código de los modelos de los ADCs usados en los modelos *float* y coma fija se encuentran en el anexo A.

2.4. Comparativa de modelos y resultados

En este capítulo se ha mostrado la metodología para simular y emular convertidores de potencia utilizando lenguajes de descripción de *hardware*. Este apartado muestra una comparativa de todos los modelos de simulación propuestos, teniendo en cuenta diferentes parámetros, como tiempo de simulación, precisión, resolución, etc. Hasta ahora se ha supuesto que, en ausencia de problemas de resolución, los resultados serían lo suficientemente realistas como para ser útiles a la hora de diseñar reguladores. Sin embargo, los modelos propuestos son simplificaciones de las plantas reales, puesto que no simulan las no idealidades de los convertidores así como pérdidas eléctricas. Por tanto, este apartado realiza no sólo comparaciones entre los diferentes modelos propuestos, sino también con resultados experimentales mediante un prototipo de convertidor de potencia. De esta forma, se puede demostrar si los modelos de las plantas son suficientemente precisos para su utilización en el diseño de reguladores. Salvo que se especifique otras condiciones, las simulaciones presentan los parámetros

indicados en la tabla 2.1, teniendo en cuenta que la carga usada es resistiva, con una potencia de 300 W cuando la tensión de salida es 400 V.

La simulación mixta analógico digital se ha probado con el simulador SystemVision 5.7 de Mentor Graphics. Las simulaciones VHDL de los modelos *real*, *float* y coma fija se han realizado con la herramienta Modelsim 6.5b de Mentor Graphics. Por su parte, las implementaciones de los modelos *float* y coma fija se han implementado en una FPGA Xilinx XC3S1000-4FT256.

El primer criterio para comparar los diferentes sistemas de pruebas es si estos sistemas son solamente **simulables** o si también son **emulables**, es decir, si se pueden sintetizar e implementar en una FPGA para así depurar el lazo completo dentro de ella, que es lo que se denomina HIL. La ventaja principal de la emulación es su mayor velocidad respecto a la simulación, como se verá más adelante. El modelo mixto analógico-digital y el modelo con señales *real* solamente pueden ser simulados, mientras que los modelos con señales *float* y coma fija pueden ser tanto simulados como emulados.

Otro criterio de comparación es el **esfuerzo de diseño**, es decir, la complejidad de diseño que suponen. Teniendo esto en cuenta, la simulación mixta es la más sencilla ya que permite implementar el sistema realizando un esquemático gráfico, con la técnica de «arrastrar y soltar», no necesitando describir a mano el convertidor de potencia. Por otra parte, los modelos con señales *real* y *float* necesitan codificación en VHDL, pero se pueden realizar transformando directamente las ecuaciones del modelo a VHDL, sin preocuparse por el ancho de las señales o resoluciones. El modelo en coma fija requiere codificación en VHDL y tener en cuenta el ancho, formato y resolución de cada señal del modelo, aumentando la complejidad del modelo. Sin embargo, se presupone que el diseñador conoce con soltura el uso de VHDL y coma fija, ya que el controlador seguramente esté realizado en VHDL y utilizando señales de coma fija. Por último, el modelo en coma fija usando *sfixed* tiene una complejidad de diseño igual al coma fija sin ayuda de bibliotecas, ya que el diseñador debe determinar el ancho de cada señal para que la relación precisión/frecuencia sea óptima. Sin embargo, la implementación en *sfixed* es más fácil, haciendo que la dificultad global de este método esté en un punto intermedio entre coma fija sin bibliotecas y *float*.

Uno de los criterios de comparación fundamentales es el **tiempo de simulación**. Este capítulo aborda la simulación y emulación de sistemas que requieren simulaciones largas, bien por su complejidad (como puede ser la simulación de un procesador embebido) o bien porque se deben simular millones de ciclos de reloj. En particular, en el sistema propuesto para PFC, el periodo de reloj de la FPGA es de 10 ns, mientras que el tiempo de estabilización del lazo de corriente es de 109 ms. Por

Tabla 2.4: Resultados de tiempo simulando 200 ms.

Sistema	Simulación/Emulación	Tiempo de simulación	Aceleración
Simulación mixta	Simulación	2 h 13 m 21 s 751 ms	
Tipo *real*	Simulación	2 m 14 s 646 ms	59,4x
Tipo *float*	Simulación	2 h 5 m 14 s 438 ms	1,1x
Tipo *float*	Emulación	3 s 228 ms	2 478,9x
Coma fija	Simulación	2 m 24 s 871 ms	55,2x
Coma fija	Emulación	277 ms	28 887,2x
Coma fija (*sfixed*)	Simulación	29 m 30 s 780 ms	4,5x
Coma fija (*sfixed*)	Emulación	294 ms	27 216,2x

tanto, para ver el comportamiento del regulador durante un transitorio hace falta simular decenas de millones de ciclos de reloj.

En la tabla 2.4 se muestra el tiempo que tardan los diferentes sistemas propuestos en realizar una simulación de 200 ms. Esta simulación de 200 ms permite simular un transitorio, pero si se quieren simular varios escalones en la carga del convertidor, por ejemplo, sería necesario simular varios segundos. Aunque los sistemas con señales *float* y coma fija están pensados para ser emulados, también se han probado en simulación para completar la tabla de tiempos. Las simulaciones se han realizado en un ordenador Intel Core 2 Duo E6550 a 2,33 GHz y con 4 GB de memoria RAM. Los tiempos de emulación se han extraído teniendo en cuenta la frecuencia máxima de reloj que permite su implementación en una FPGA de bajo coste Xilinx XC3S1000, como se mostrará más adelante. Así, se conseguiría una emulación en tiempo real si la frecuencia máxima del sistema fuera 100 MHz. La emulación en tiempo real se consigue con esa frecuencia ya que el tiempo de integración del modelo del convertidor es de 10 ns, es decir, el inverso de 100 MHz.

En la tabla 2.4 también se muestra la aceleración conseguida en cada sistema en comparación con la simulación mixta. Como se puede observar, los sistemas emulados son significativamente más rápidos que los simulados, obteniendo aceleraciones de 28 887,2x usando coma fija, y de 2 478,9x usando señales *float*. El sistema más rápido es el que usa coma fija, pero su diseño, como se ha comentado, es más complejo. El modelo *float* es 10 veces más lento que el de coma fija en emulación, pero su diseño es más sencillo. En cualquier caso, la aceleración por emulación en *float* debería ser suficiente en casi cualquier aplicación, ya que se consiguen emulaciones de segundos.

Teniendo en cuenta solamente los tiempos de simulación, los tipos *real* y coma fija son más de 50 veces más rápidos que la simulación mixta y *float*, requiriendo simulaciones de minutos en vez de horas. La simulación con tipo *float* es casi tan lenta como la simulación mixta debido a que el *hardware* necesario para implementar operaciones en coma flotante es muy complejo y, por tanto, también lo es su simulación. Debido a esta razón, la simulación del tipo *float* carece de sentido. Es notable la diferencia entre la simulación de coma fija con y sin biblioteca *sfixed*. Como se

puede observar, esta biblioteca no está optimizada para su simulación, por lo que la velocidad de su simulación está un orden de magnitud por debajo de la simulación de la coma fija estándar. Sin embargo, en emulación los dos modelos en coma fija se implementan en *hardware*, siendo los dos prácticamente idénticos, por lo que la velocidad de emulación es similar.

Los sistemas emulables (*float* y coma fija) también pueden ser comparados por el **área** que ocupan en *hardware*. El sintetizador XST, que está integrado en las herramientas de Xilinx ISE 12.3, no es capaz de compilar el paquete float_pkg. Por ello se ha usado el sintetizador Synplify Premier E2011 de Synopsys. En cambio, el modelo en coma fija solamente usa tipos estándar, así que se ha sintetizado con ambas herramientas. La tabla 2.5 muestra la frecuencia máxima y los recursos ocupados en ambos modelos cuando son implementados en una FPGA Xilinx XC3S1000. En particular, se han desarrollado tres tipos de síntesis:

- El modelo del convertidor elevador o *boost* sin incluir el controlador, ya que es la única parte que cambia entre los dos sistemas.
- El sistema completo HIL, el cual incluye el modelo y controlador, pero sin incluir la implementación de analizadores digitales.
- El sistema completo HIL y un analizador digital ChipScope para depurar el sistema, el cual será explicado en el apartado 2.4.1.

Como muestra la tabla, el modelo en coma fija utiliza muchos menos recursos de la FPGA en comparación con el modelo *float*, y además permite una frecuencia máxima 10 veces más alta aproximadamente. La razón es que los sumadores y multiplicadores en coma flotante son mucho más complejos que los necesarios en el modelo en coma fija. Por tanto, si el área es una opción importante en la simulación a realizar, el sistema emulable a elegir es el de coma fija. El modelo en coma fija con *sfixed* muestra resultados similares, tanto en área como en frecuencia, al modelo en coma fija sin bibliotecas. Esto es debido a que el diseño y elección de ancho de cada señal ha sido idéntico.

Teniendo en cuenta solamente el modelo en coma fija, su síntesis con la herramienta XST produce sistemas hasta un 30 % más rápidos que usando Synplify. Esto es debido a que el sintetizador decide usar más multiplicadores embebidos (MULT18x18) que Synplify, utilizando las opciones por defecto. Con esta configuración por defecto se ha observado que Synplify usa LUTs para realizar multiplicaciones por una constante, usando más LUTs y reduciendo la frecuencia máxima de funcionamiento. Usando

Tabla 2.5: Recursos ocupados en la FPGA (Xilinx XC3S1000) según el modelo.

Sistema	Frec. máxima	LUTs de 4 entradas	Flip flops	Mult. 18x18	BRAMs (16 kB)
Modelo *boost*: coma fija (sintetizador XST)	68,747 MHz	170	60	2	0
Modelo *boost*: coma fija (sintetizador Synplify)	61,584 MHz	380	79	0	0
Modelo *boost*: coma fija *sfixed* (sintetizador XST)	67,944 MHz	145	60	2	0
Modelo *boost*: tipo *float* (sintetizador Synplify)	6,103 MHz	7 355	76	0	0
HIL: coma fija (sintetizador XST)	68,781 MHz	447	361	4	1
HIL: coma fija (sintetizador Synplify)	56,497 MHz	658	358	1	1
HIL: coma fija *sfixed* (sintetizador XST)	67,810 MHz	448	365	4	1
HIL: tipo *float* (sintetizador Synplify)	6,085 MHz	9 332	392	1	1
HIL con CS: coma fija (sintetizador XST)	72,202 MHz	814	665	4	24
HIL con CS: coma fija (sintetizador Synplify)	55,121 MHz	1 036	662	1	24
HIL con CS: coma fija *sfixed* (sintetizador XST)	72,106 MHz	816	688	4	24
HIL con CS: tipo *float* (sintetizador Synplify)	6,196 MHz	9 412	685	1	24

directivas se puede ordenar al sintetizador Synplify que maximice el uso de multiplicadores embebidos, aunque requiere que el diseñador tenga conocimientos sobre el uso de la herramienta.

Como se ha comentado anteriormente, los sistemas emulados serían sistemas de tiempo real si consiguieran una frecuencia de 100 MHz, ya que esa es la frecuencia de integración del modelo del convertidor de potencia. Por tanto, los sistemas emulados propuestos no son de tiempo real. En cualquier caso, es importante remarcar que el controlador, ya sin modelo del convertidor, está diseñado para funcionar a 100 MHz y en el sistema en producción funcionará a dicha frecuencia.

Otra de los criterios fundamentales para diferenciar los sistemas propuestos es la **precisión** de los mismos. Si la precisión de un sistema no es suficientemente alta, ninguna mejora en la aceleración podría compensarlo. Por tanto, se presentan dos experimentos para comparar los sistemas.

En el primer experimento se han simulado todos los sistemas en lazo cerrado usando el controlador para corrección de factor de potencia, extrayendo el valor en régimen

permanente de g_{in} (G_{in}), es decir, de la salida del regulador del lazo de tensión. Se ha elegido G_{in} porque este parámetro está afectado tanto por el cálculo de la tensión de salida como de la corriente de entrada, comprobándose el correcto funcionamiento de los modelos propuestos para ambos lazos. Si la tensión de salida tuviera imprecisiones, el lazo de tensión modificaría g_{in} para corregir el error en la tensión. Por otra parte, si hubiera errores en el cálculo de la corriente de entrada, no habría balance entre las potencias de entrada y salida, y esta inestabilidad también sería compensada modificando g_{in}. Por tanto, el valor G_{in} es un buen parámetro para comprobar la precisión de los modelos propuestos.

La tabla 2.6 muestra los resultados del primer experimento, comparándolos con la G_{in} ideal. Este valor ideal puede ser calculado con la fórmula $G_{in} = \frac{P}{V_g^2}$ [28], la cual representa a un modelo sin pérdidas. La simulación mixta no representa un modelo sin pérdidas, sino que incluye elementos parásitos. Para compensar estas pérdidas, el G_{in} en la simulación mixta es 1,7 % mayor que el ideal. El resto de modelos no modelan pérdidas y por esta razón se comparan con la G_{in} ideal. El modelo con señales *real*, el cual usa coma flotante de doble precisión (64 bits) alcanza el valor más preciso de G_{in}, con un error del 0,31 %, seguido por el modelo en coma fija, el cual obtiene un error del 0,38 %. El modelo basado en señales *real* no sufre problemas de resolución en el cálculo de las variables de estado ya que usa 64 bits, y como se verá en el apartado 2.6.2, no son necesarios tantos bits para los cálculos. Por tanto, el error del 0,31 % se debe a las no idealidades que tiene la implementación del regulador. Por una parte, se limita el ciclo de trabajo del PWM hasta el 98 % para que la frecuencia de conmutación sea constante y evitar sobrecorriente durante los transitorios, y también se limita el ciclo de trabajo mínimo al 1 % para conseguir frecuencia constante de conmutación. Otra de las no idealidades se debe a que se han añadido los modelos de los ADC que incluyen retrasos y erorres de cuantización no despreciables usando solamente 12 bits. La diferencia entre el error de sistema *real* y el sistema en coma fija (0,07 %) sí puede achacarse a problemas leves de resolución. Aunque el cálculo interno de la corriente y la tensión es muy preciso, se truncan dichos valores para realimentar el lazo, originando pequeños problemas de resolución. Añadiendo más bits a dicha realimentación se podrían solucionar fácilmente estos problemas.

El modelo con señales *float* obtiene un gran error en G_{in}: 9,66 %. Esto es debido a que este modelo usa señales de 32 bits, para intentar reducir los recursos necesarios para su implementación, y estas señales no tienen suficiente resolución para almacenar los incrementos de i_L y v_{out}. El problema de resolución será tratado ampliamente en el apartado 2.6. En cualquier caso, debido a los problemas de resolución, el sistema *float* de 32 bits no es válido por los errores que produce. Una posible solución es

Tabla 2.6: Precisión de los modelos usados como convertidores para PFC.

Sistema	Simulación/ Emulación	G_{in}	Error en G_{in} en comparación con el G_{in} ideal
G_{in} ideal		0,00567108	
Resultados experimentales		0,00564575	$-0{,}45\,\%$
Simulación mixta	Simulación	0,00576782	$1{,}71\,\%$
Tipo *Real*	Simulación	0,00565338	$0{,}31\,\%$
Tipo *float*	Sim/Emulación	0,00512314	$9{,}66\,\%$
Coma fija	Sim/Emulación	0,00564957	$0{,}38\,\%$
Resultados obtenidos en régimen permanente con la referencia de V_{out} igual a 400 V			

usar señales *float* de 64 bits, ampliando enormemente el área necesaria y reduciendo drásticamente la frecuencia del sistema. Otra solución sería aumentar el tiempo de integración Δt, para que los valores incrementales sean mayores y así evitar los problemas de resolución. Sin embargo, la precisión del sistema es inversamente proporcional a la magnitud del parámetro Δt, por lo que un aumento del último afecta negativamente a la precisión del sistema. En el ejemplo propuesto, Δt es igual a 10 ns y la frecuencia de conmutación es de 100 kHz, por lo que la resolución del ciclo de trabajo que se conecta al transistor es del 0,1 %. Si en cambio el parámetro se cambiara a 100 ns, la resolución del ciclo de trabajo descendería hasta el 1 %. La conclusión es que el modelo *float* no es apropiado para sistemas con altas frecuencias de conmutación, es decir, aquellos en los que Δt sea pequeña.

La tabla 2.6 también muestra el G_{in} obtenido en las pruebas experimentales. El valor de G_{in} de los resultados experimentales debería ser ligeramente mayor que la ideal debido a las pérdidas eléctricas que se originan, sin embargo, se puede observar que es menor. Esto es debido a imprecisiones en las etapas de medición. Por ejemplo, si la ganancia de los conversores analógico digitales no es exactamente igual a la ganancia esperada, el parámetro G_{in} del regulador diverge, siendo mayor o menor. En este caso el valor experimental de G_{in} es $0,45\,\%$ menor que lo experado debido a estos errores en la medición. Una de las conclusiones importantes es que la precisión de los modelos que se han presentado es incluso mayor que los errores de medición que se producen en condicionaes experimentales. Por tanto, queda demostrada la precisión en régimen permanente de los modelos en coma fija y *real*. El modelo en coma fija usando la biblioteca *sfixed* no ha sido añadido a la tabla, ya que obtiene exactamente los mismos resultados que el modelo en coma fija a mano, siempre que los tamaños de los registros sean idénticos.

El segundo experimento que se ha realizado para comprobar la precisión de todos los sistemas propuestos es introducir un escalón en la carga conectada al convertidor elevador. De esta forma, se puede comprobar la respuesta dinámica de cada modelo. En la figura 2.10 se puede observar el comportamiento de v_{out} en los diferentes

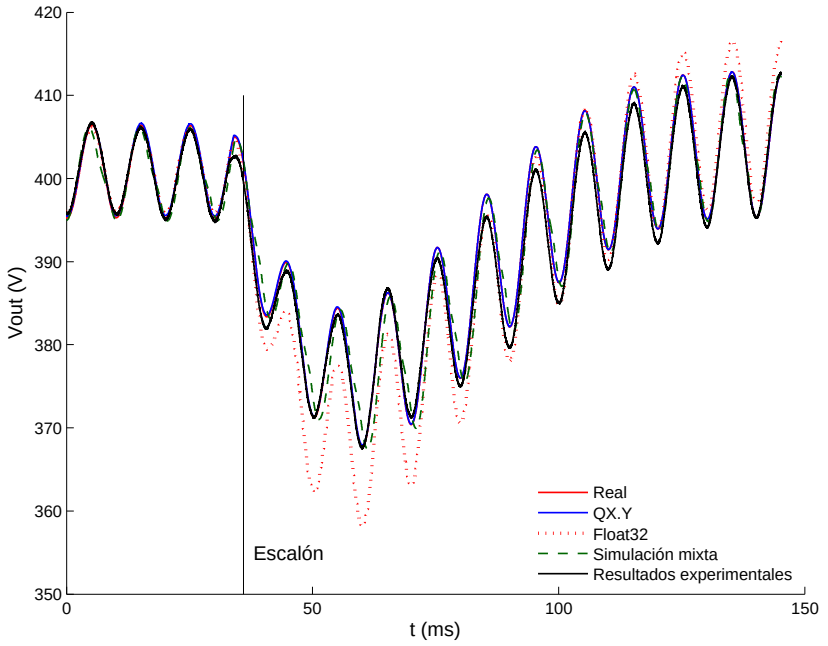


Figura 2.10: Comparación de los modelos propuestos ideales y con pérdidas tras un escalón en la carga de 1176 Ω a 540 Ω ($V_{out}Ref = 400\ V$).

sistemas cuando se produce un escalón en la carga resistiva desde 1176 Ω a 540 Ω cuando el sistema estaba previamente en régimen permanente. Nuevamente se han introducido resultados experimentales para compararlos con los resultados de los modelos propuestos. Los modelos *real* y coma fija tienen una respuesta muy similar ya que ambos tienen suficiente precisión, pero ninguno de ellos modelan las pérdidas eléctricas. Por último, se puede observar que el modelo *float* es aparentemente más amortiguado no habiendo razón aparente para ello. El problema realmente es su escasa resolución. Los problemas de resolución y su influencia en la precisión se verán en la sección 2.6. Todos los sistemas se comportan de forma casi idéntica en régimen permanente (figura 2.11) dado que están funcionando en lazo cerrado y el controlador se encarga de obtener la misma v_{out} en todos los casos.

Por último, se ha realizado otro experimento para comprobar la precisión de los modelos. Las figuras 2.12 y 2.13 muestran las formas de ondas de la corriente de entrada en régimen permanente, tanto en el caso del prototipo real como en la simulación del modelo real. Los resultados con el modelo en coma fija no se muestran ya que son similares al comportamiento del modelo *real*. Las formas de onda del prototipo presentan más ruido, pero tanto las formas de onda experimentales como las del modelo *real* tienen un comportamiento muy parecido. En la figura 2.12 el convertidor está en condiciones nominales y el regulador de corriente es el óptimo, es decir, el presentado en la tabla 2.2. En este caso, las formas de ondas experimental y simulada tienen el

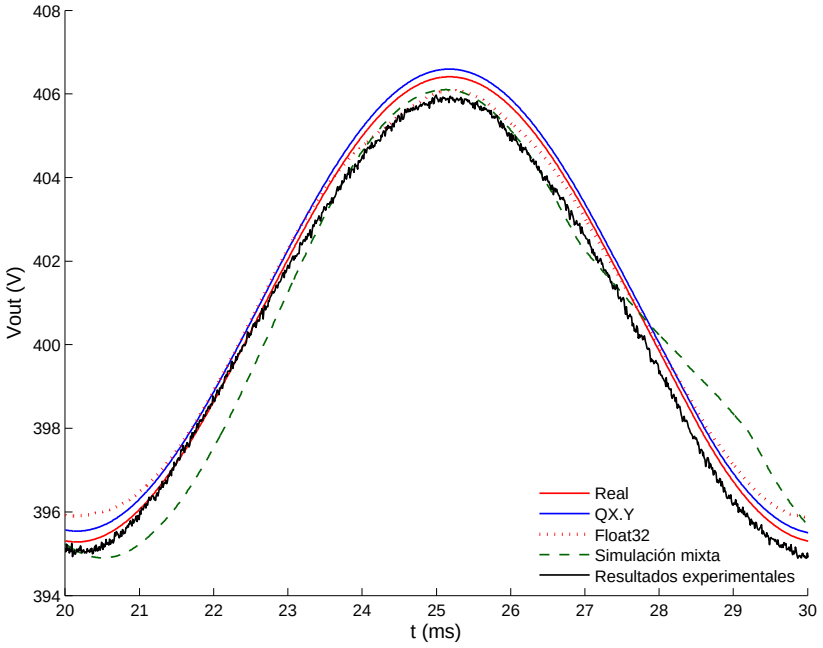


Figura 2.11: Ampliación de la figura 2.10 (parte izquierda en régimen permanente).

pico en la corriente de entrada localizado $0,5\ ms$ antes del punto ideal, que debería ser $5\ ms$, es decir, un semiciclo de red. El factor de potencia en condiciones nominales es 0,9967 para el prototipo y 0,9964 para la simulación *real*, siendo prácticamente idénticos. Por su parte, la figura 2.13 muestra el mismo experimento, pero utilizando un regulador no ideal, el cual tiene un cuarto de ganancia respecto al regulador de corriente presentado en la tabla 2.2. En esta figura se puede ver claramente que la corriente de entrada no es ideal, pero nuevamente tanto los resultados del prototipo como los de simulación son similares. La conclusión que podemos extraer de este experimento es que las técnicas de simulación que se han propuesto en este capítulo son válidas para evaluar el rendimiento de reguladores antes de ser implementados en una prueba experimental. De esa forma, podremos extraer formas de onda realistas, y sacar conclusiones sobre el funcionamiento en lazo cerrado, como puede ser el factor de potencia.

2.4.1. Emulación y extracción de información

En los apartados anteriores se han abordado el diseño e implementación del sistema de pruebas. Cuando se ha terminado con la implementación, el sistema debe ser simulado o emulado para así extraer información relevante.

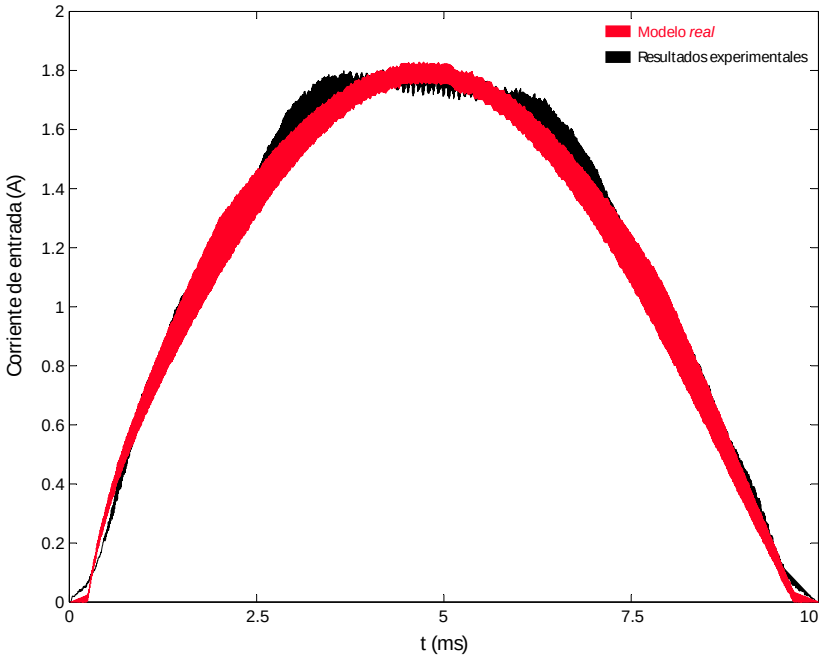


Figura 2.12: Formas de onda de la corriente de entrada en régimen permanente en el prototipo y en el modelo *real* con el regulador de corriente óptimo.

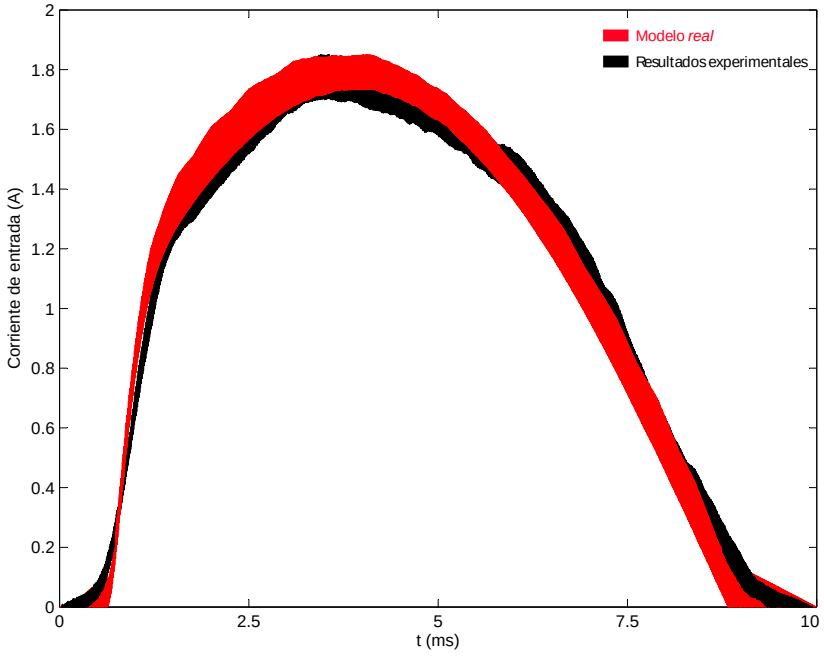


Figura 2.13: Formas de onda de la corriente de entrada en régimen permanente en el prototipo y en el modelo *real* con el regulador de corriente con un cuarto de ganancia respecto al óptimo.

En el caso de la **simulación** mixta analógica digital, la extracción de valores es **sencilla** ya que el propio simulador contiene herramientas de visualización de señales analógicas o también las señales modeladas en VHDL. La simulación de los modelos en VHDL igualmente es sencilla debido a que estos simuladores permiten mostrar todas las señales del sistema y gráficas en caso de que representen valores de naturaleza analógica, tales como la tensión o corriente.

La extracción de datos en los **sistemas emulados** (*float* y coma fija) tiene mayor interés debido a que no se puede realizar de forma directa, debido a que el sistema se ejecuta en lazo de cerrado en el interior de una FPGA y no dentro de un ordenador. Una posibilidad es rutar los valores que se desean visualizar a **pines de salida en una FPGA**, y capturar la información mediante un analizador lógico o conducirlos a un conversor digital analógico. Otra posibilidad es **implementar en la propia FPGA un analizador lógico** el cual transmita los datos necesarios a un ordenador mediante, por ejemplo, el cable de programación JTAG.

Si se desea usar FPGAs de la compañía Xilinx, se puede añadir un analizar lógico en el diseño llamado ChipScope. La arquitectura de este analizador puede constar de varios elementos. Sin embargo, la arquitectura más sencilla es utilizar un analizador ILA (del inglés *Integrated Logic Analyzer*) y un controlador ICON (del inglés *Integrated CONtroller*). El primero se encarga de muestrear un número limitado de señales externas o internas del diseño implementado, mientras que el segundo componente es la interfaz entre el analizador y un programa que se ejecuta en un ordenador. Una vez que se dispara un *trigger*, el analizador ILA toma un número determinado de muestras en las señales deseadas y las guarda en BRAMs. Se puede acceder a estas BRAMs desde el módulo ICON, transfiriendo su información al ordenador para que el ingeniero de pruebas las pueda visualizar, ver figura 2.14.

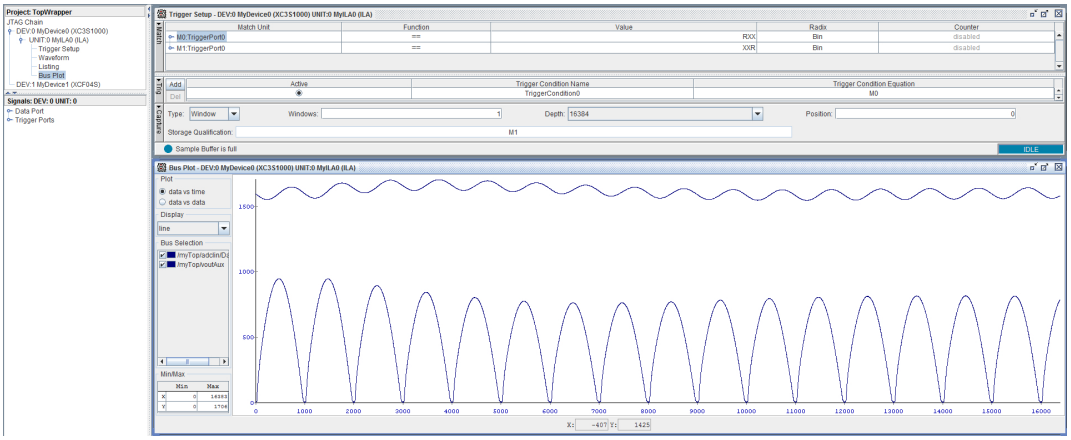


Figura 2.14: Captura del analizador Xilinx ChipScope.

Tabla 2.7: No idealidades añadidas al modelo.

Elemento	Característica	Valor
Puente de diodos	Tensión de codo en conducción (v_B)	1,14 V
Bobina	Resistencia en serie (R_L)	0,6965 Ω
Interruptor	Resistencia de conducción del MOSFET (R_{ON})	0,3 Ω
Diodo	Tensión de codo en conducción (v_D)	1,3 V

El número de BRAMs de la FPGA es limitado, especialmente en las FPGA de bajo coste. Aunque depende de la FPGA empleada y del tamaño de la información que se desea extraer, un número razonable de muestras a tomar es de 16 000 muestras. Si el analizador digital tomara muestras a la velocidad del reloj de la FPGA, se obtendrían los datos correspondientes al 1,6 % de un semiciclo de red. Para obtener un tiempo de representación mayor, se puede ampliar el periodo de muestreo del analizador. Por ejemplo, tomando muestras cada 1 000 ciclos de reloj, obtendríamos 1 000 muestras por semiciclo de red durante 16 semiciclos de red (160 ms), suficiente para ver la evolución de un transitorio en el lazo de tensión (el cual tiene un tiempo de estabilización de aproximadamente 100 ms), como se muestra en la figura 2.14.

2.5. Influencia de las pérdidas en el modelo

Los modelos en lenguaje HDL presentados en este capítulo carecen de la representación de pérdidas eléctricas. En el capítulo se ha primado la claridad de los modelos, suponiendo que las pérdidas eléctricas no aportan gran información a la simulación del convertidor junto al regulador. Sin embargo, un análisis más exhaustivo debe realizarse para apoyar esta premisa. En esta sección se propone la adición en los modelos de las pérdidas de primer orden de los componentes usados en el convertidor. Igualmente se muestran los resultados obtenidos con esta modificación y se valora la conveniencia de añadirlas al modelo. Los parámetros que se añaden están reflejados en la tabla 2.7. Los valores indicados son los reales en el convertidor usado durante este capítulo, extraídos de las hojas de datos de los componentes utilizados, salvo en el caso de la resistencia serie de la bobina. Dicho valor ha sido medido, ya que la bobina se ha hecho a mano.

La principal desventaja de añadir estas pérdidas eléctricas es que el modelo se complica ligeramente, haciendo que el diseño sea algo más costoso y que la simulación y emulación sean algo más lentas. El comportamiento de un convertidor elevador teniendo en cuenta las pérdidas descritas viene definido por el siguiente sistema de ecuaciones:

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (v_g - v_B - v_{R_L} - v_{R_{ON}}) \\ v_{out}(k) &= v_{out}(k-1) - \frac{\Delta t}{C} \cdot i_R \end{aligned} \tag{2.20}$$

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (v_g - v_{out} - v_D - v_{R_L}) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{2.21}$$

$$\begin{aligned} i_L(k) &= 0 \\ v_{out}(k) &= v_{out}(k-1) - \frac{\Delta t}{C} \cdot i_R \end{aligned} \tag{2.22}$$

En las ecuaciones anteriores, v_{R_L} es la caída de potencial debida a la resistencia en serie de la bobina R_L, y $v_{R_{ON}}$ es la caída de potencial debida a la resistencia en conducción del MOSFET R_{ON}. La metodología para implementar este sistema de ecuaciones es similar a la descrita durante este capítulo, por lo que el código final sólo se mostrará en el anexo A.

Se ha añadido el cálculo de las pérdidas para los modelos *real* y coma fija. El modelo *float* no se ha modificado ya que se demostró anteriormente que sus problemas de resolución no permiten la simulación del convertidor propuesto. Se han repetido los experimentos mostrados anteriormente, teniendo en cuenta las pérdidas. En el primer experimento se ha extraído el valor de G_{in} del regulador cuando hay pérdidas y se ha comparado con el mostrado anteriormente. La tabla 2.8 muestra los resultados obtenidos. Como se puede observar, el parámetro G_{in} cambia un 2,024 % en el caso del modelo *real* y 1,148 % en el caso del modelo en coma fija. G_{in} varía al añadir pérdidas ya que al añadir pérdidas, la potencia de entrada tiene que ser mayor para conseguir la misma potencia de salida. En ambos casos, al añadir las pérdidas, el valor de G_{in} se aproxima al obtenido con la simulación mixta. Este comportamiento es el esperado, ya que la simulación mixta también modela pérdidas.

También se ha repetido el experimento para comprobar la dinámica de los modelos, aplicando un transitorio. La figura 2.15 muestra el transitorio para el modelo en coma fija con pérdidas, además del modelo sin pérdidas y los resultados experimentales. Como se puede observar, la influencia de las pérdidas en la dinámica del modelo es

Tabla 2.8: Precisión de los modelos usados como convertidores para PFC.

Sistema	Simulación/ Emulación	G_{in}	Error en G_{in} en comparación con el G_{in} ideal
Simulación mixta	Simulación	0,00576782	1,71 %
Tipo *Real* sin pérdidas	Simulación	0,00565338	−0,31 %
Tipo *Real* con pérdidas	Simulación	0,00576782	1,1 %
Coma fija sin pérdidas	Sim/Emulación	0,00564957	−0,38 %
Coma fija con pérdidas	Sim/Emulación	0,00571442	0,76 %
Resultados obtenidos en régimen permanente con la referencia de V_{out} igual a 400 V			

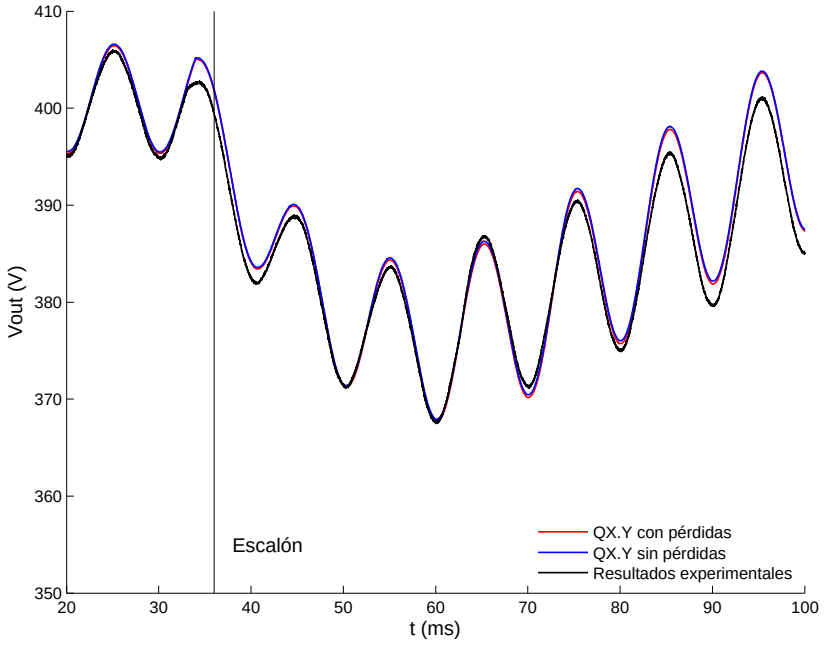


Figura 2.15: Comparación de los modelos propuestos ideales y con pérdidas tras un escalón en la carga de 1176 Ω a 540 ω en el instante 10 ms ($v_{out}Ref = 400\ V$).

prácticamente nula, no aportando información relevante. La dinámica del sistema depende básicamente de la inductancia de la bobina de entrada y la capacidad del condensador de salida, L y C, aparte de la ganancia del regulador, que es la misma en todos los casos. Dado que los modelos con y sin pérdidas tienen el mismo regulador y los mismos valores de C y L, la dinámica es similar en todos ellos. Sin embargo, sí hay más diferencias entre cualquier modelo planteado y los valores experimentales, ya que los valores reales de C y L no son iguales a los teóricos, debido a las tolerancias de los componentes. A priori este hecho no se puede corregir en una simulación, por lo que la simulación indica el comportamiento *medio* de los distintos convertidores reales si se fabrica más de uno. Por tanto las pérdidas no aportan gran información para el comportamiento dinámico.

2.6. Análisis de resolución

En el anterior apartado se han probado modelos de simulación y emulación usando diferentes tipos de señales. Se analizaron parámetros como el esfuerzo de diseño, el tiempo de simulación, el área ocupada en emulación o la precisión. Referente a la precisión, se comentó que el modelo *float* no era válido para los parámetros de diseño escogidos, dado que sus señales no tenían suficiente resolución para almacenar correctamente incrementos pequeños en las variables de estado.

Una posible solución es aumentar el ancho de las señales internas de cálculo, pero el área ocupada aumenta y la frecuencia de cálculo se reduce. Otra posibilidad es aumentar el tiempo de integración del modelo del convertidor, Δt. De esa forma, los incrementos en las tensiones y corrientes son mayores, y la resolución necesaria es menor. Sin embargo, esto deteriora la precisión de la integración y disminuye la resolución del PWM en la simulación, como se verá a continuación.

El objetivo de este apartado es definir los parámetros de diseño que influyen en la resolución del sistema, y analizar el ancho necesario de las señales para que estas puedan almacenar con suficiente resolución las variables de estado.

2.6.1. Resolución del ciclo de trabajo del PWM

El convertidor de potencia depende de un transistor el cual es controlado mediante una señal PWM. Esta señal PWM procede del lazo de corriente del regulador, como se comentó en el apartado 2.2.1. A la hora diseñar el módulo de PWM, hay que tener en cuenta principalmente dos parámetros: la frecuencia de conmutación y la resolución del ciclo de trabajo. La frecuencia de conmutación depende de la aplicación que se desee desarrollar. Así, para convertidores de varios kW, se suele utilizar frecuencias de decenas de kHz, mientras que para convertidores menores a un kW, se suelen aplicar frecuencias de cientos o miles de kHz. Por otra parte, la resolución del ciclo de trabajo es un parámetro de diseño que determina la exactitud de la respuesta del regulador. Resoluciones altas permiten que la tensión de salida del convertidor se ajuste correctamente al valor deseado, mientras que resoluciones bajas dificultan dicho ajuste, incluso originándose problemas como el ciclo límite [29, 30].

La frecuencia de conmutación y la resolución del ciclo de trabajo del PWM determinan el tiempo de integración (Δt) máximo que la simulación debe tener. Si el ciclo de trabajo del PWM tiene una resolución de res_{PWM} bits, y su frecuencia de conmutación es f_{SW}, el modelo debe comprobar si el PWM ha cambiado su valor al menos cada Δt segundos:

$$\Delta t = \frac{1}{f_{SW} \cdot 2^{res_{PWM}}} \tag{2.23}$$

Aplicando la fórmula 2.23, el tiempo máximo de integración con el convertidor y regulador propuestos en apartados anteriores es $\Delta t = \frac{1}{100kHz \cdot 1000} = 10\ ns$. La resolución en nuestro caso es igual a 1 000, dado que se usan 10 bits, pero solamente se usan 1 000 valores de los 1 024 que ofrecen los 10 bits.

La fórmula 2.23 determina el máximo tiempo de integración para que el modelo del convertidor aproveche toda la resolución del PWM. Sin embargo, el tiempo Δt podría ser aumentado a costa de perder resolución en la simulación y por tanto precisión. Por cada vez que se duplique el tiempo Δt equivale a ignorar el bit menos significativo del ciclo de trabajo y, por tanto, la resolución del PWM se reduce a la mitad. La pregunta es cuánto podría ser aumentado ese tiempo sin deteriorar en exceso la precisión de la simulación.

Se han realizado pruebas de simulación para comprobar la resolución mínima que debe tener el PWM. Para ello, se ha utilizado el sistema completo de simulación utilizando diferentes anchos de palabra en el ciclo de trabajo. La precisión del sistema se ha comprobado con el parámetro G_{in}, el cual es un buen indicador de la precisión del sistema, como se comentó en el apartado 2.4. La tabla 2.9 y la figura 2.16 muestran los resultados obtenidos. La figura muestra dos umbrales de error en G_{in}: un error del 10 % se considera intolerable, y un error del 5 % es el máximo recomendable para realizar simulaciones. Los resultados obtenidos muestran una tendencia exponencial, por lo que no tendría ningún sentido usar menos de 7 u 8 bits, pero tampoco usar más de 10 bits. Usando los umbrales definidos, se podrían usar 8 bits obteniendo un error considerable, aunque **lo recomendable sería usar 9 o 10 bits para el ciclo de trabajo**. Es importante resaltar que los resultados obtenidos coinciden con los controladores de PWM que se encuentran en el estado del arte para este tipo de aplicaciones [31, 32].

El experimento se ha realizado usando el sistema en coma fija, y modificando el número de bits utilizados en la salida del lazo de corriente, es decir, en el ciclo de trabajo. En los resultados originales del modelo en coma fija, el cual usaba 10 bits de ciclo de trabajo, se obtuvo un error en G_{in} igual al 0,38 %, mientras que en el experimento que se acaba de mostrar el error con 10 bits de ciclo de trabajo es igual al 0,90 %. Esto es debido a que en el primer experimento se usó la técnica del *dither* [30] para aumentar internamente la resolución del ciclo de trabajo pero usando los mismos bits externos. Sin embargo, para observar directamente los efectos

Tabla 2.9: Precisión del sistema en relación a la resolución del ciclo de trabajo.

Bits del ciclo de trabajo	G_{in}	Error en G_{in} en comparación con el G_{in} ideal
10 bits (1 000 valores)	0,00572205	0,90 %
9 bits	0,005867	3,45 %
8 bits	0,0061264	8,029 %
7 bits	0,00664139	17,11 %
6 bits	0,00772858	36,28 %
5 bits	0,0100327	76,91 %

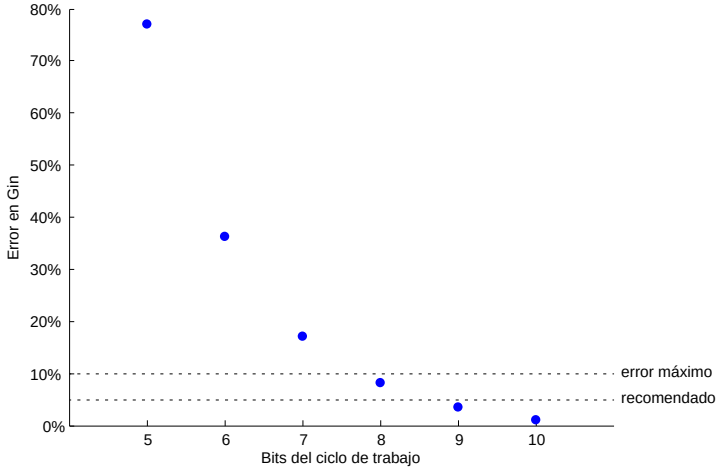


Figura 2.16: Precisión del sistema en relación a la resolución del ciclo de trabajo.

del número de bits en el ciclo de trabajo, se ha eliminado esta técnica para realizar este experimento.

2.6.2. Resolución de las señales internas de cálculo

En el anterior apartado se ha comentado cómo afecta la resolución del PWM al tiempo de integración del modelo del convertidor. Cuanto mayor sea esta resolución, menor debe ser el periodo de integración. Por su parte, este periodo afecta al ancho necesario en las variables de estado, dado que a menor tiempo de integración, menores los incrementos a guardar, y por tanto, se necesita mayor resolución.

El ancho necesario para una señal viene determinado por la magnitud de los valores a guardar en ella. El ancho aumenta según aumenta la diferencia de órdenes de magnitud entre los valores de una señal y sus incrementos. En las variables de estado es crítico, porque hay que almacenar su valor anterior y sumarle el pequeño incremento del nuevo paso de integración, es decir, hay que tener suficiente resolución tanto en la variable (x) como en su incremento (Δx). En el resto de variables vale con tener suficiente resolución en su valor (x), así que no suele haber problemas de

resolución. Por ejemplo, los valores que vienen del ADC, como v_g, suelen estar en 12 bits y es suficiente, ya que expresan un valor en un instante de tiempo y no se usan para integrarlo.

En el caso de la tensión de salida, la tensión en el convertidor propuesto es de unos cientos de voltios mientras que los incrementos son del orden de micro voltios. Para guardar un incremento Δx en una señal con valor x, se necesita un mínimo de bits determinado por la ecuación 2.24:

$$bits_x = \lceil \log_2 \frac{x}{\Delta x} \rceil + n \tag{2.24}$$

donde n es el número de bits para expresar el incremento. Cuando $n = 1$, esta ecuación indica el mínimo número de bits necesarios para guardar x y Δx. De esta forma Δx se expresaría con un '1' en el bit menos significativo. Sin embargo, no siempre los incrementos serán igual a Δx, sino que varían ligeramente. Si no se añaden más bits, solamente se podrían incrementar múltiplos de Δx, obteniendo una resolución muy baja. Por tanto, n debe ser mayor que uno. De nuevo la pregunta es cuántos bits son necesarios para expresar los incrementos.

En el modelo propuesto se calculan dos variables de estado: la tensión de salida y la corriente de entrada. A modo de ejemplo, se calcularán cuántos bits son necesarios para el cálculo de la tensión de salida. Se ha escogido este cálculo porque es la presenta más problemas de resolución en nuestro caso.

La tensión de salida está determinada por la ecuación 2.6. Por tanto, la tensión depende de Δt, C, i_C y de su valor anterior. El valor de C está determinado por el diseño del convertidor, y Δt se calcula usando el procedimiento del apartado anterior. i_C depende del estado del transistor, pero suponiendo que el transistor está cerrado, los incrementos típicos de la tensión de salida están expresados en la ecuación 2.25:

$$\Delta v_{out} = \frac{\Delta t}{C} \cdot i_R = \frac{10\ 10^{-9}\ s}{100\ 10^{-6}\ F} \cdot 0{,}75\ A = 7{,}5\ 10^{-5}\ V \tag{2.25}$$

La ecuación 2.25 muestra el incremento en la tensión de salida cuando el transistor está cerrado o en general si el modo de conducción es DCM. Para la ecuación se ha tenido en cuenta que la potencia es igual a 300 W y por tanto i_R es igual a 0,75 A. Aunque este resultado depende del estado del convertidor y del transistor, lo que se

Tabla 2.10: Precisión del sistema en relación a n (número de bits del incremento de v_{out}).

Bits del incremento de v_{out}	G_{in}	Error en G_{in} en comparación con el G_{in} ideal
12 bits	0,00564957	0,38 %
11 bits	0,00565338	0,31 %
10 bits	0,00567627	0,09 %
9 bits	0,00569534	0,43 %
8 bits	0,00572586	0,97 %
7 bits	0,00590134	4,06 %
6 bits	0,00601959	6,15 %
5 bits	0,00631714	11,39 %
4 bits	0,00765228	34,94 %
3 bits	0,0089035	57 %

desea es calcular el orden de magnitud de los incrementos, así que se da este valor como un valor "medio".

Sabiendo los incrementos, los bits necesarios para expresar v_{out}, teniendo en cuenta que la referencia de tensión es igual a 400 V están definidos en 2.26:

$$bits_{Vout} = \lceil \log_2 \frac{400\ V}{7{,}5\ 10^{-5}\ V} \rceil + n = 23 + n \tag{2.26}$$

v_{out} debe expresarse en, al menos, 24 bits. Sin embargo, como se ha comentado, el incremento de la tensión debe expresarse con más de 1 bit para expresarse con precisión. Para comprobar cómo afecta el número de bits del incremento en la precisión general del sistema, se han hecho pruebas de simulación con diferentes anchos en los incrementos de la tensión de salida, es decir, con diferentes valores de n. La tabla 2.10 y la figura 2.17 muestran los resultados obtenidos en el experimento. De nuevo se han establecido los umbrales de error máximo tolerable y error máximo recomendable, que se establecen con un error del 10 % y 5 % respectivamente. La gráfica resultante de nuevo tiene un comportamiento exponencial, deteriorándose en gran medida la precisión cuando n es igual a cuatro, y no obteniendo apenas mejora en la precisión cuando se usan más de 8 bits. Según los umbrales, se deben usar al menos 7 bits para obtener un error tolerable, y **se recomienda usar 7 u 8 bits para el incremento** de la tensión de salida.

El estudio sobre el número de bits necesarios en las señales internas se ha realizado sobre el cálculo de la tensión de salida. Sin embargo, también se puede calcular cuántos bits son necesarios en el cálculo de la corriente de entrada. Para ello, hay que extraer nuevamente los valores típicos de la corriente, y sus incrementos. Nuevamente suponemos que el transistor está cerrado para calcular los incrementos típicos, (2.27):

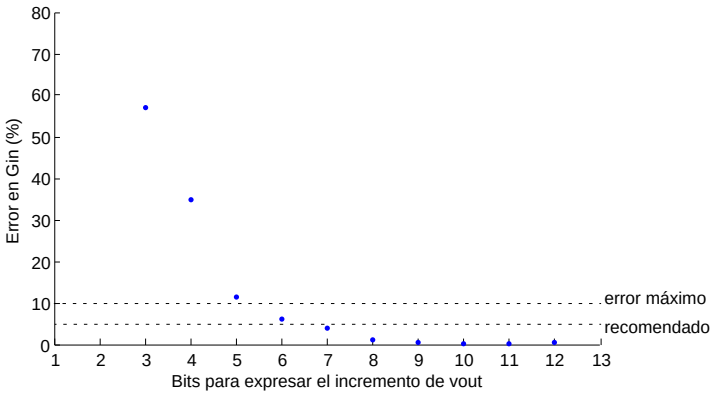


Figura 2.17: Precisión del sistema en relación a n (número de bits del incremento de v_{out}).

$$\Delta i_L = \frac{\Delta t}{L} \cdot v_g = \frac{10 \cdot 10^{-9}}{5 \cdot 10^{-3}} \cdot 230 = 4{,}6 \cdot 10^{-4}\ A \tag{2.27}$$

Para calcular el número de bits necesarios para el cálculo de la corriente de entrada también necesitamos saber los valores típicos de la misma. Dado que las potencias de entrada y salida deben ser iguales, se puede hacer cumplir la ecuación (2.28):

$$P_{entrada} = P_{salida}$$

$$V_g \cdot I_L = V_{out} \cdot I_R$$

$$I_L = \frac{V_{out} \cdot I_R}{V_g} = \frac{400 \cdot 0{,}75}{230} = 1{,}3\ A \tag{2.28}$$

Sabiendo los valores típicos de I_L y sus incrementos, el número de bits necesarios para el cálculo de la corriente de entrada está definido por la ecuación (2.29):

$$bits_{I_L} = \lceil \log_2 \frac{1{,}3}{4{,}6 \cdot 10^{-4}} \rceil + n = 12 + n \tag{2.29}$$

Como se puede observar, el cálculo de la corriente requiere un menor número de bits $(12 + n)$ que el número de bits necesarios para el cálculo de la tensión de salida

$(23 + n)$, y por esta razón se han realizado las pruebas con diferentes anchos de palabra para el cálculo de la tensión de salida.

2.6.3. Guía para elegir los parámetros de simulación

En este apartado se presenta una guía y resumen de los parámetros de simulación que anteriormente se han comentado. Con esta guía se pretende que el diseñador tenga una idea aproximada del ancho necesario para las señales de la simulación. De esa forma, si el diseñador decide usar señales en coma fija, podrá dimensionarlas correctamente, y si el diseñador opta por utilizar el tipo de datos *float*, podrá ver con antelación si este tipo de señal es válida en la aplicación deseada.

En primer lugar, se debe decidir el periodo de integración de la simulación para poder aprovechar toda la resolución del PWM:

1. Se obtiene la frecuencia de conmutación del PWM. Esta frecuencia no es un parámetro de simulación, sino que viene impuesta por el diseño del regulador.
2. Se obtiene la resolución del ciclo de trabajo del PWM. Igualmente este parámetro se elige en la etapa de diseño del regulador, aunque debe ser de, al menos, 8 bits (recomendándose 9 o 10) para conseguir una simulación suficientemente precisa.
3. Se extrae el periodo de integración máximo para la simulación resolviendo la ecuación: $\Delta t = \frac{1}{f_{SW} \cdot 2^{res_{PWM}}}$

Una vez extraído el tiempo máximo de integración para la simulación, se puede calcular el ancho necesario para las señales de cálculo:

1. A través de las ecuaciones en diferencias, se obtienen los incrementos típicos de la señal que se desea analizar.
2. El ancho necesario para x se calcula con la ecuación: $bits_x = \log_2 \frac{x}{\Delta x} + n$, donde n es el número de bits para expresar el incremento Δx. Un tamaño razonable para expresar el incremento es 8 bits.

Usando esta guía, se puede comprobar que el modelo en *float* planteado en este capítulo no tiene suficiente resolución en la señal que representa la tensión de salida. Esto es debido a que su tiempo de integración es igual a 10 ns, y para utilizar

8 bits para expresar los incrementos de la tensión de salida, harían falta 31 bits significativos, es decir, de mantisa. Sin embargo, el estándar de coma flotante de precisión simple, *float32*, solamente ofrece 24 bits de mantisa (23 almacenados y un '1' fijo), que equivale a tener 1,65 bits para expresar los incrementos de tensión de salida. Este tipo de datos ha sido usado previamente para simulación de convertidores de potencia [13, 14]. Sin embargo, las frecuencias de conmutación de estos sistemas son mucho menores de las propuestas en este ejemplo. Por tanto, el uso de *float32* carece de sentido para simulaciones de sistemas de alta frecuencia de conmutación.

2.6.4. Pruebas sistemáticas de análisis de resolución

En esta sección se ha abordado cómo decidir el tamaño que deben tener las variables de estado de los modelos propuestos. El método mostrado se basa en el cálculo del tamaño necesario para almacenar valores representativos de las variables. Este proceso heurístico ha sido probado mediante pruebas sistemáticas para comprobar la idoneidad de los tamaños obtenidos.

El método sistemático de pruebas se basa en simular el modelo del convertidor para un conjunto acotado de tamaños de las variables de estado. También se han probado diferentes condiciones de operación, tales como tensiones de entrada, salida, transitorios, etc. Dado el gran número de pruebas necesarias, se ha utilizado un entorno de pruebas sistematizadas llamado OVM (del inglés *Open Verification Methodology*) [33]. Este entorno es flexible y permite automatizar las pruebas propuestas. La arquitectura del sistema OVM se muestra en la figura 2.18. El modelo a probar, llamado DUV (*Design Under Verification*), está conectado al entorno de pruebas mediante un *driver* y un monitor. El primer elemento convierte las señales de entrada del modelo a probar desde un formato fácil de procesar (por ejemplo, coma flotante), hasta el formato requerido por el modelo, es decir, coma fija con formato QX.Y. Por su parte, el monitor realiza la conversión inversa, traduciendo de coma fija a un formato cómodo para el análisis de los resultados.

En el nivel más alto hay otros dos elementos, el secuenciador y el puntuador. El primero genera los estímulos necesarios para el modelo: tensión de entrada, ciclos de trabajo del interruptor y la corriente demandada por la carga. El puntuador se encarga de comprobar los resultados del modelo y generar los reportes pertinentes. Los resultados deben compararse con una referencia para medir el error generado por el modelo en coma fija. Si lo que se desea comprobar es la influencia del tamaño de las variables de estado en la precisión del modelo, sería incorrecto comparar el modelo de coma fija con resultados experimentales. Esto es debido a que el modelo en coma

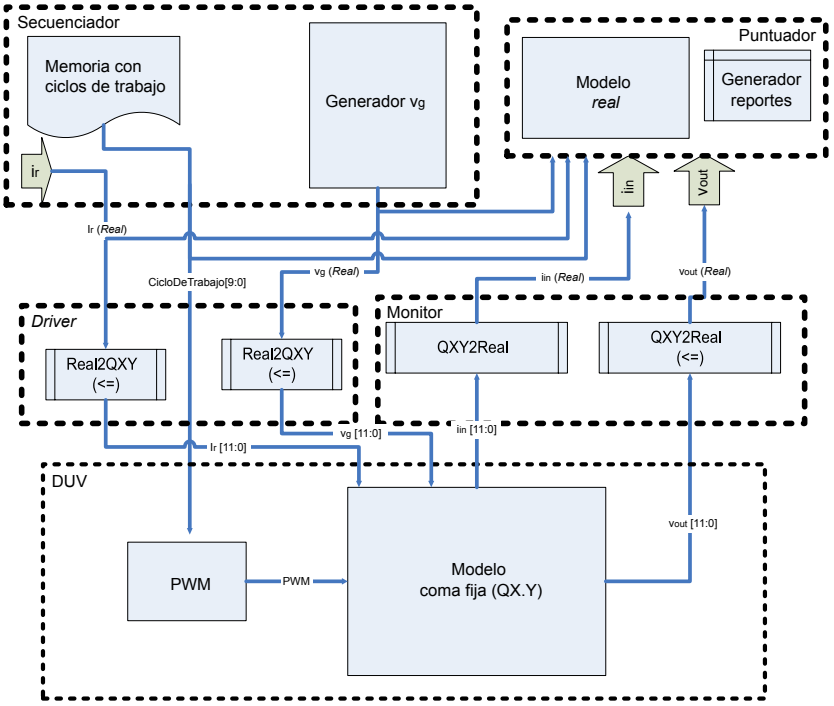


Figura 2.18: Arquitectura OVM usada para las pruebas sistematizadas de resolución en las variables de estado.

fija no tiene en cuenta ciertas no idealidades. Por tanto, para comprobar el efecto del tamaño de las variables, se debe comparar dos modelos similares donde únicamente cambien los tamaños de sus señales. Teniendo en cuenta que el modelo *real* usa coma flotante de 64 bits, éste no presentará problemas de resolución. Por ello, el modelo *real* puede usarse como modelo de referencia para estas pruebas.

Los escenarios en los que se ha probado el convertidor vienen detallados en la tabla 2.11. Se han probado cuatro escenarios en los que se ha variado el valor de la bobina, condensador y la relación de tensiones de entrada y salida, y por tanto la potencia demandada. El primer escenario coincide con el convertidor propuesto durante todo el capítulo. Sin embargo, se han realizado pruebas sobre más escenarios para comprobar si realmente son importantes los elementos particulares de cada convertidor a la hora de elegir el tamaño de las variables de estado. Con la misma finalidad, para cada escenario se han probado diferentes tipos de carga: resistiva (usada durante todo el capítulo), de potencia y de corriente.

Para cada escenario se han realizado pruebas sistemáticas variando el ancho de las variables que almacenan la tensión de salida y la corriente de entrada. N_v indica el tamaño de la señal que representa la tensión de salida, mientras que N_i fija el tamaño

Tabla 2.11: Escenarios en los que se ha probado el modelo del convertidor)

	L	C	V_g	V_{out}	P_{out}
Escenario 1	5 mH	100 μF	230 V	400 V	300 W
Escenario 2	1 mH	100 μF	230 V	400 V	300 W
Escenario 3	1 mH	100 μF	110 V	300 V	150 W
Escenario 4	1 mH	470 μF	230 V	400 V	300 W

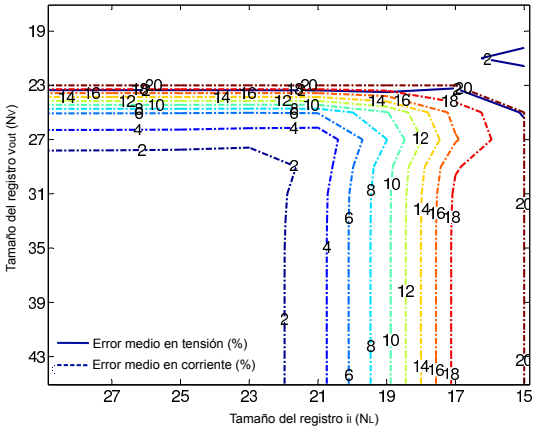


Figura 2.19: Escenario 1

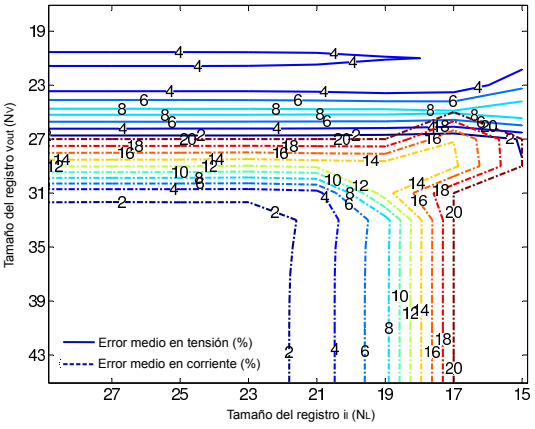


Figura 2.20: Escenario 2

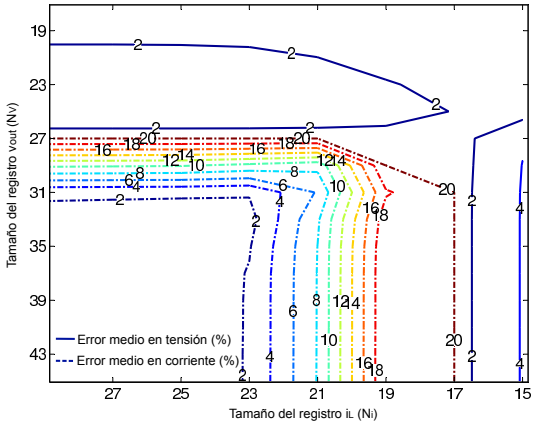


Figura 2.21: Escenario 3

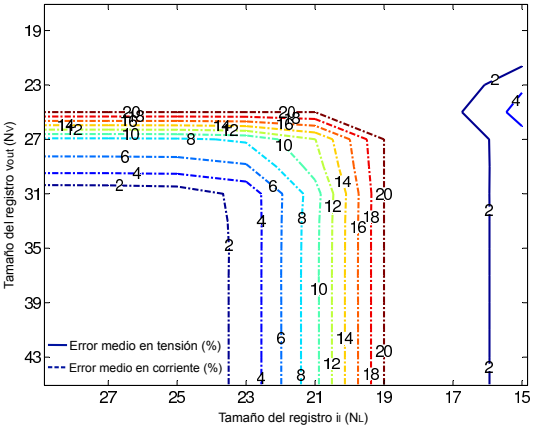


Figura 2.22: Escenario 4

Figura 2.23: Errores en la tensión de salida y corriente de entrada según N_v y N_i para carga resistiva.

de la señal de la corriente de entrada. Ambos parámetros han sido variados entre 16 y 47 bits.

Los resultados de la batería de pruebas pueden encontrarse en la figura 2.23. En particular, se muestran los cuatro escenarios para carga resistiva, para poder ser comparados con los valores heurísticos calculados anteriormente. Como puede observarse, el error en el registro de la corriente de entrada siempre es mayor que el error en el registro de la tensión de salida, independientemente del tamaño de ambos registros. Las

figuras también muestran claramente que el registro de la tensión de salida necesita un tamaño mayor que el registro de la corriente de entrada.

En el apartado 2.6.3 se señaló que el número recomendado de bits reservado para el incremento de una variable es 8. De esa forma, el tamaño total recomendado para v_{out} era igual a 31 y el tamaño final recomendado para i_L era igual a 20. Teniendo en cuenta los tamaños descritos, en la figura 2.19 se puede observar que el error en la corriente es aproxidamente igual al 6 %, mientras que el error en la tensión es menor al 2 %. Estos errores son tolerables, pero pueden reducirse aumentando el tamaño de los registros. Si se sumara un bit a los registros, el error en corriente decrecería hasta el 4 % y el error en tensión se mantendría por debajo del 2 %.

Este proceso de prueba sistemática requiere cientos de simulaciones (distintos anchos de cada señal, además de distintas condiciones en la simulación). Lo que se propone no es llevar a cabo este largo proceso en cada diseño sólo para decidir el ancho de las variables de estado, sino que se pretende demostrar que el método heurístico propuesto en esta tesis es válido. Tras todo este proceso de pruebas sistemáticas, el resultado es prácticamente el mismo que el propuesto en esta tesis. De esta forma se ha demostrado la validez del método heurístico mostrado anteriormente para la elección de tamaños de registros.

2.7. Conclusiones

En este capítulo se han mostrado diferentes posibilidades de simulación de reguladores digitales para convertidores de potencia. Los reguladores deben ser simulados en su etapa de diseño utilizando herramientas específicas. Sin embargo, en la etapa de codificación en HDL se pueden introducir errores que podrían provocar daños al convertidor o incluso personales. Además, la implementación real presenta no idealidades tales como retrasos en los ADC, ancho de palabra limitado, limitación en el ciclo de trabajo máximo y mínimo, etc. Por tanto, la simulación del regulador en su estado de codificación final es realmente importante.

Este capítulo ha mostrado diferentes modos de realizar simulaciones y emulaciones. La simulación mixta permite diseñar un sistema de pruebas de forma sencilla, añadiendo componentes gráficos en un esquemático, además del regulador diseñado en HDL. Su gran inconveniente es la lentitud de las simulaciones resultantes, llegando a horas si se debe simular un sistema complejo.

La otra gran alternativa es modelar el convertidor de potencia también en HDL. Dependiendo del tipo de datos que se use para definir el modelo, la velocidad de

Tabla 2.12: Comparación de las posibilidades de simulación/emulación descritas.

Sistema	Emulable	Esfuerzo de diseño	Tiempo de simulación	Tiempo de emulación	Área ocupada	Precisión
Sim. mixta	No	Poco	Horas	-	-	Alta
Real	No	Poco	Minutos	-	-	Alta
Float32	Sí	Poco	Horas	Segundos	Mucha	Baja
Coma fija	Sí	Bastante	Minutos	Milisegundos	Poca	Alta*
Coma fija *sfixed*	Sí	Medio	Bastantes minutos	Milisegundos	Poca	Alta*
* Con elección correcta de resolución						

simulación y el esfuerzo en el diseño varía. El tipo de datos *real*, que es un tipo en coma flotante, permite modelar el convertidor de forma sencilla, obteniendo simulaciones de minutos. El tipo *float* provee también señales en coma flotante, pero además permite ser sintetizado, por lo que el sistema completo puede emularse, con enormes aceleraciones en el tiempo empleado en la prueba. El gran inconveniente de *float* en 32 bits es que carece de suficiente resolución para modelar convertidores de potencia con alta frecuencia de conmutación. Además, otro problema del tipo *float* es el gran área que ocupa cuando se implementa para emularlo, por lo que el uso de *float64* no es razonable. Igualmente se debe señalar que el tipo de datos *float* tiene escasa compatibilidad con los sintetizadores HDL que se encuentran en el mercado. En el capítulo se ha presentado también un modelo con señales en coma fija. Este modelo requiere un mayor esfuerzo de diseño, pero permite emulación con tiempos de alrededor de milisegundos. Por último, se ha propuesto el uso de la biblioteca *sfixed* para la implementación en coma fija. El uso de esta biblioteca no ayuda en el diseño del modelo, pero facilita la implementación HDL del modelo, por lo que su uso es razonable.

La tabla 2.12 compara los diferentes sistemas presentados, teniendo en cuenta los principales criterios que los diferencian.

El diseño en coma fija se ha realizado eligiendo los anchos de palabra adecuados para obtener suficiente resolución en cada variable, incrementando su ancho en las señales que deben integrarse, y reduciendo los anchos en las señales que realimentan el modelo. Además de la comparativa de los diferentes sistemas, la principal aportación de este capítulo son las transformaciones que se han aplicado a las ecuaciones en diferencias, las cuales permiten acelerar notablemente la frecuencia de trabajo del modelo, así como el *hardware* necesario para su implementación en el sistema emulable.

En el capítulo también se ha mostrado cómo afecta añadir pérdidas eléctricas de primer orden a los modelos. Se ha visto que los modelos HDL con pérdidas se acercan

al comportamiento de la simulación mixta en régimen permanente, ya que la última también incluye pérdidas eléctricas. En las pruebas de dinámica, no se ha visto un gran aporte de información al añadir las pérdidas. La dinámica del sistema depende principalmente del regulador (invariable en todos los modelos), y de la capacidad del condensador de salida (C) y de la inductancia de la bobina de entrada (L). Por tanto, todos los modelos, con y sin pérdidas, tienen un comportamiento similar en las pruebas de dinámica. Y por otra parte, los resultados experimentales varían ligeramente de los anteriores, ya que sus valores de C y L son diferentes a los valores teóricos, debido a las tolerancias en dichos valores, variables en cada convertidor real. Por tanto, los modelos con pérdidas, así como la simulación mixta, muestran el comportamiento *medio* de los convertidores reales que puedan ser fabricados con dichos parámetros.

Capítulo 3

Corrección de factor de potencia mediante el precálculo de los ciclos de trabajo

3.1. Introducción

La corrección de factor de potencia lleva estudiándose y aplicándose desde hace numerosos años. Como se ha comentado anteriormente, la corrección de factor de potencia consigue rectificar la tensión alterna de entrada, a la vez que se regula la tensión media de salida y la corriente de entrada. Los técnicas PFC permiten reducir el contenido armónico que se produce al rectificar la tensión de entrada mediante la fuente conmutada, emulando una carga resistiva. La técnica clásica de PFC está mostrada en la figura 3.1. En la figura se muestran dos lazos de control, uno interno para regular la corriente de entrada, y uno externo para regular la tensión media de salida. El lazo externo genera un comando de potencia, para subir o bajar la tensión media. Por otro lado, el lazo interno consigue que la corriente sea sinusoidal, midiendo la corriente de entrada real y comparándola con una referencia. La referencia de corriente se puede obtener con el comando de potencia del lazo externo y la tensión de entrada. Así, el convertidor PFC consigue que la corriente de entrada sea proporcional a la tensión de entrada, emulando una carga resistiva y, por tanto, minimizando el contenido armónico.

La figura 3.2 muestra la evolución de las tensiones de entrada y salida, corriente de entrada, y potencias de entrada y salida durante dos semiciclos de red. Como se puede ver, la potencia de entrada es variable porque es el resultado de la multiplicación de

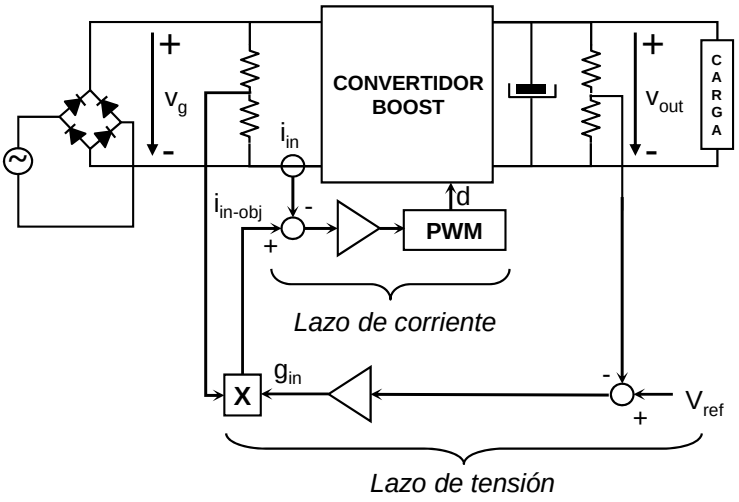


Figura 3.1: Técnica PFC con un convertidor elevador.

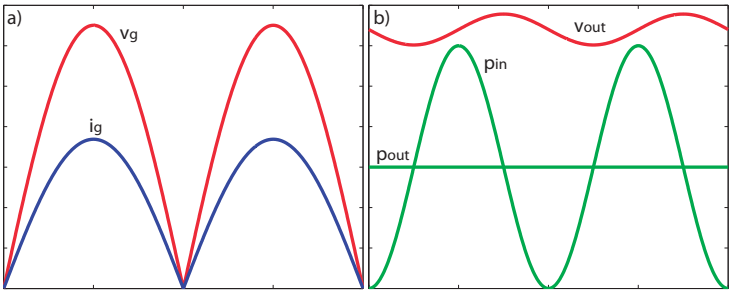


Figura 3.2: a) Corriente y tensión de entrada en un convertidor PFC. b) Potencia de entrada, y tensión y potencia de salida en un convertidor PFC.

dos ondas sinusoidales en fase (v_g e i_{in}). Sin embargo, la potencia de salida debe ser aproximadamente constante. Por tanto, incluso en régimen estacionario hay un rizado inevitable en la tensión de salida, debido a que la potencia de entrada varía al doble de la frecuencia de la red eléctrica.

Como se ha comentado, el lazo externo mide la tensión de salida, mientras que el lazo interno mide la tensión de entrada y la corriente de entrada. Por tanto, las técnicas clásicas de corrección de factor de potencia miden tres parámetros analógicos. La medición de señales analógicas nunca es deseable, ya que incrementan el coste del convertidor, además de añadir otros inconvenientes. Especialmente, el sensado de la corriente de entrada no es una tarea trivial, ya que hay que llegar a un compromiso entre coste, pérdidas eléctricas, precisión y ancho de banda. Sin embargo, el uso de reguladores digitales hace más factible evitar alguna medida.

Este capítulo muestra diferentes técnicas PFC, las cuales hacen uso de **ciclos de trabajo precalculados**. En vez de calcular en tiempo real el estado en el que debe

estar el interruptor del convertidor, según las mediciones de las señales analógicas, se aplicará al interruptor una señal almacenada previamente en una memoria. De esta forma, el regulador **idealmente consistiría en una memoria y un sistema de sincronización con la tensión de entrada**, eliminando el sensado de tensiones y corrientes. El sistema de sincronización se puede conseguir fácilmente con un comparador de tensión de bajo ancho de banda, comparando la tensión de entrada con un umbral, y sin necesidad de usar un ADC para esta tarea. Como el sistema descrito funcionaría en lazo abierto, siendo altamente vulnerable ante cambios de las condiciones de entrada (especialmente tensión de entrada y potencia demandada), se deben añadir sistemas de control en lazo cerrado. Por ello, se ha añadido únicamente un ADC para controlar la tensión de salida. Este ADC nos permite conocer la tensión media de salida, requisito necesario en un conversor, pero también el rizado de dicha tensión, el cual es indicativo de la potencia demandada por la carga. En el capítulo se muestran diferentes técnicas para realizar dicha regulación **usando únicamente un ADC**, siendo ésta la principal **aportación original de este capítulo de tesis**.

Para poder realizar las técnicas de regulación sobre ciclos de trabajo precalculados, se mostrará un profundo análisis del ciclo de trabajo en un convertidor PFC. El ciclo de trabajo puede ser dividido en diferentes componentes y tratarse de forma diferente en la etapa de regulación. De esta forma y haciendo uso del ADC, el sistema es capaz de actuar ante cambios de la tensión de entrada y de la carga del convertidor.

El capítulo muestra cómo realizar el precálculo del ciclo de trabajo, y muestra tres técnicas de regulación sobre ciclos precalculados. Las técnicas están descritas en orden creciente de precisión pero también de complejidad. Sin embargo, todas ellas comparten las mismas necesidades en cuanto a componentes: **un único ADC y la sincronización con la red eléctrica**, usando un comparador de tensión para esta tarea. Por último se ofrece una comparativa de todas las técnicas propuestas, en términos de factor de potencia, armónicos obtenidos, cumplimiento de normativas y recursos *hardware* requeridos.

3.1.1. Factor de potencia y distorsión armónica

El factor de potencia es una medida que se utilizará durante este capítulo para valorar la calidad de los sistemas propuestos. El factor de potencia o PF (del inglés *Power Factor*) es la relación entre la potencia real transmitida a la carga y la potencia aparente tomada de la red eléctrica, por lo que define la eficiencia en el convertidor de potencia [28]:

$$PF = \frac{potencia\ media\ (W)}{potencia\ aparente\ (VA)} \tag{3.1}$$

El primer término se refiere a la potencia media calculada multiplicando los valores instantáneos de tensión y corriente durante un ciclo de red, y el segundo término es el producto de los valores RMS de la corriente y tensión. Idealmente esta relación es igual a 1, indicando máxima eficiencia en la conversión, produciéndose cuando la carga (en este caso el convertidor) cumple la ley de Ohm, es decir, si el convertidor se comporta como una resistencia. Esto produce que la tensión y la corriente de entrada sean proporcionales y el contenido armónico es nulo.

Otro parámetro usado en la medición de la eficiencia de un convertidor es la distorsión armónica total o THD (del inglés *Total Harmonic Distortion*) [28]. El THD se define como la relación entre el valor RMS de una señal obviando la componente fundamental y el valor RMS de la componente fundamental. Así, el THD cuando no hay componente continua es:

$$THD = \frac{\sum_{n=2}^{\infty} I_n^2}{I_1} \tag{3.2}$$

En la corrección de factor de potencia es habitual medir la distorsión armónica de la corriente de entrada, ya que este parámetro indica inequívocamente la calidad en la conversión de energía, siendo nulo en el caso ideal.

3.1.2. Estado del arte

Los primeros correctores de factor de potencia diseñados usaban reguladores analógicos, opción que sigue siendo mayoritaria. Las ventajas del control digital son numerosas, tanto en el caso de corrección de factor de potencia, como en otro tipo de circuitos de potencia. Algunas de las ventajas son la versatilidad en la regulación, reducción en número y tamaño de elementos pasivos, reguladores más robustos ante el envejecimiento y calor, gestión y modificación del regulador en tiempo real, etc. Por otra parte, entre las desventajas están el mayor coste de desarrollo, la limitación del ancho de banda debido al proceso de muestreo y su mayor coste de fabricación. En particular, el coste de fabricación cada vez se está reduciendo más, no implicando

grandes limitaciones en este sentido. En cuanto a la limitación del ancho de banda, progresivamente se han ido presentado técnicas que solucionaban este perjuicio.

Inicialmente se demostró que la corrección de factor de potencia se podía realizar usando reguladores digitales en microcontroladores [34] y en DSPs (del inglés *Digital Signal Processor*) [35, 36]. Posteriormente, se fueron añadiendo diferentes mejoras que añadían funcionalidades al corrector o bien reducían su coste. Así, algunas propuestas mejoran el lazo de corriente [31, 37]. En [31] se muestra una de las primeras implementaciones digitales de un lazo de control de carga, gracias al uso de FPGAs en vez de DSPs. Además, en este artículo se propone la idea de realizar el lazo de tensión controlando la tensión de salida máxima cada semiciclo de red. De esa forma, miden la máxima tensión de salida que hay en cada ciclo de conmutación, obteniendo un filtrado, implícito en la medición, del rizado de la tensión de salida al doble de la frecuencia de red.

Otras propuestas permiten aumentar el ancho de banda del lazo de tensión [38, 39, 40, 41, 42, 43]. Normalmente el lazo de tensión tiene un ancho de banda muy bajo, alrededor de 10 a 20 Hz. Este bajo ancho de banda es debido a que mayores anchos harían que el lazo de tensión intentara corregir el rizado de la tensión de salida, interfiriendo la corrección de factor de potencia y siendo contraproducente. El problema es que el citado lazo es muy lento ante transitorios en la tensión de salida. En [38, 40] se mejora el ancho de banda del lazo de tensión, estimando el rizado de la tensión de salida, y restándolo a la tensión real de salida, obteniendo una medida sin rizado. En [39] se muestran mejoras similares en el lazo de tensión para aumentar su ancho de banda. Además, muestra la ventaja, posible en controladores digitales, de medir la corriente de forma sincronizada con el PWM del interruptor, consiguiendo un filtrado implícito de la medida. En [41, 42] también se aumenta el ancho de banda del lazo de tensión, eliminando la influencia del rizado de la tensión de salida, esta vez aplicando un filtro en peine, el cual rechaza la frecuencia del rizado y la de sus múltiplos.

Como se ha comentado, uno de los elementos que incrementan el coste del convertidor es el sensado de las señales analógicas. Las técnicas tradicionales de corrección de factor de potencia requieren medir tres parámetros: tensión de entrada, corriente de entrada, y tensión de salida. Numerosos artículos han presentado métodos para eliminar mediciones en estas variables. En [44] se presenta un sistema de alimentación ininterrumpida que mide únicamente la tensión de salida y la corriente de salida. [45] elimina la medida de la tensión de entrada introduciendo un observador de perturbaciones. En [46] también se elimina la medición de la tensión de entrada, realizando una estimación de la misma. [32] implementa un convertidor DNLC (del inglés *Digital*

Non-Lineal Carrier) en el que no se mide la tensión de entrada. En [47] se presenta un método para la estimación de la tensión de salida a través de la medición de la corriente de entrada. [48, 49] presentan un método para medir las señales analógicas sin ADCs, usando señales generadas en forma de diente de sierra, y utilizando comparadores.

Como se ha comentado anteriormente, la etapa de sensado de corriente es costosa, no sólo en correctores de factor de potencia. Por ello, no es de extrañar que numerosas propuestas hayan mostrado métodos para reducir las etapas de sensado.

En [50, 51] se muestran resúmenes de las técnicas más habituales de medición de la corriente. Una práctica habitual es utilizar un sensor resistivo (*shunt*), para medir con un ADC la caída de tensión en la resistencia. La ventaja es la simplicidad y el bajo coste del *shunt*. Los principales inconvenientes son que la resistencia genera pérdidas eléctricas y que el calor generado debe evacuarse. Además, cabe destacar que las pérdidas eléctricas son proporcionales al cuadrado de la corriente que proporciona el convertidor. Por otra parte, para disminuir las pérdidas, el *shunt* proporciona una tensión muy pequeña, por lo que debe amplificarse antes de ser cuantificada por un ADC, por lo que el coste se incrementa.

Es importante destacar que la corriente de entrada tiene una frecuencia igual a la frecuencia de conmutación, estando alrededor de las decenas o cientos de kiloherzios. De esta forma, el ADC que mide la corriente de entrada debe tener un ancho de banda suficientemente alto para sensar esta señal. Aunque en el mercado se pueden encontrar ADCs con prestaciones mucho mayores, un ADC de muy bajo coste no puede usarse. Por otra parte, en el caso de las tensiones de salida y entrada, el ADC puede ser de muy baja frecuencia de muestreo, ya que estos dos parámetros tienen una frecuencia igual a la frecuencia de la tensión rectificada ($100\ o\ 120\ Hz$ dependiendo el país donde se use el convertidor).

Algunas de las técnicas usan la estimación de la corriente usando medidas de tensión con ADCs [52, 53]. En particular, estas propuestas fueron aplicadas a convertidores multifase dc-dc. Por otra parte, [54] presenta un método de estimación de la corriente midiendo la caída de tensión en la bobina de un convertidor dc-dc. Aunque esa idea no es nueva, [54] propone un método de auto-ajuste el cual sirve para calibrar la medición de la corriente de entrada. El auto-ajuste se realiza provocando un escalón conocido de corriente y comparando la respuesta medida con la esperada.

En aplicaciones de corrección de factor de potencia, también se han presentado numerosos trabajos donde no se mide la corriente de entrada. Por ejemplo, en [55] se presenta un convertidor PFC en el que no se mide la corriente de entrada, pero

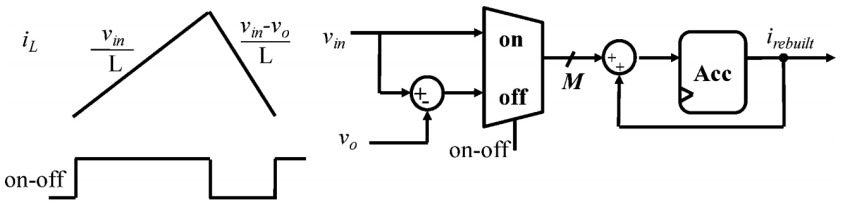


Figura 3.3: Sistema propuesto en [59].

sí se estima su paso por cero midiendo las tensiones de entrada y de salida. Usando también las medidas de las tensiones de entrada y de salida, en [56, 57, 58, 59] se presenta un método para la reconstrucción completa, no sólo su paso por cero, de la corriente de entrada. La figura 3.3 presenta la técnica usada, en la cual se miden las dos tensiones, y se estima la corriente conociendo el valor de la inductancia y el estado del interruptor. Con un acumulador, un multiplexor y dos sumadores, además de la medida de ambas tensiones, el sistema permite reconstruir la corriente de entrada. Un problema que se produce es que inevitablemente se genera error en la estimación de la corriente, y este error es acumulativo a lo largo del semiciclo de red. Para evitar este error creciente, en [60] se propone la detección del momento en el que convertidor entra en modo de conducción discontinua, y se implementa un lazo de bajo ancho de banda, el cual corrige estas desviaciones en la estimación de la corriente. Los mismos autores presentan en [61, 62], basándose en las propuestas anteriores, se centran en el análisis de los parásitos, retrasos en las conmutaciones y otras no idealidades.

[63] muestra el diseño de un sensor analógico-digital capaz de medir la corriente media usando únicamente dos comparadores de tensión y un filtro paso bajo. Este último filtra la salida de realimentación del sensor y se compara con la tensión que representa a la corriente. Este sensor se puede usar tanto para convertidores dc-dc, como ac-dc.

En [64, 65] se presentan también soluciones sin medición de corriente para corrección de factor de potencia, en las cuales el ciclo de trabajo se determina a partir de funciones del tipo $\frac{sen\Theta}{\Theta}$, donde Θ es el fasor que define la tensión de entrada.

Por otra parte, [66] muestra un convertidor PFC en el que únicamente se usa un lazo de tensión, midiendo las tensiones de entrada y de salida. Este sistema ofrece buen rendimiento ante condiciones de entrada nominales y ante transitorios, aunque no ante tensión de entrada distorsionada. Para evitar ese problema, en [67] se presenta una modificación del regulador, la cual mejora los resultados ante este caso, aunque aumenta la complejidad del sistema. En este artículo el ciclo de trabajo se obtiene midiendo la tensión en la inductancia y generando una referencia sincronizada con la tensión de entrada a partir de LUTs (del inglés *Look-UP Table*).

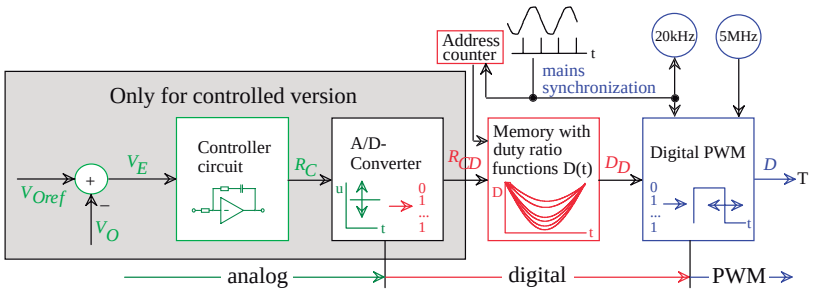


Figura 3.4: Sistema propuesto en [68, 69].

Otro método para eliminar la medida de la corriente es precalcular el ciclo de trabajo del interruptor, de forma similar a la propuesta de este capítulo. De esa forma, se puede guardar el estado del interruptor durante un semiciclo de red, y esta información se puede aplicar periódicamente, una vez que el sistema se sincronice con la tensión de entrada. Algunas de las propuestas que implementan dicha idea son [68, 69, 70, 71]. El principal problema del precálculo del ciclo de trabajo es que el factor de potencia decrece drásticamente cuando las condiciones de entrada no son exactamente iguales a las utilizadas durante el cálculo de los ciclos de trabajo.

[68, 69] presentan un método basado en ciclo precalculado, así como el control necesario para reaccionar ante cambios en la tensión de salida debidos a la carga. El control propuesto selecciona el conjunto de ciclos de trabajo más adecuado entre varios que están almacenados. Un regulador analógico se encarga de dicha tarea teniendo como entrada la tensión de salida, y generando una tensión que se discretiza con un ADC para seleccionar la memoria a usar, como se muestra en la figura 3.4. En particular, el sistema almacena ocho posibles ciclos de trabajo, limitando las posibilidades de regulación. Además, en los citados trabajos no se tienen en cuenta cambios en la tensión de entrada.

Por otra parte, [70] presenta una técnica predictiva que calcula los ciclos de trabajo para el próximo semiciclo de red, midiendo las tensiones de entrada y de salida durante el semiciclo actual. Este sistema presenta limitaciones en caso de cambios en la carga, por lo que los mimos autores mejoraron el sistema en [71]. Este último trabajo reduce dichas limitaciones, a costa de medir la corriente de entrada. Como se ve en la figura 3.5, se realizan tres mediciones: tensión de entrada, tensión de salida y corriente de entrada. Por tanto, este último método propuesto presenta una alternativa para realizar corrección de factor de potencia, pero el número de sensores no se reduce frente a una propuesta clásica.

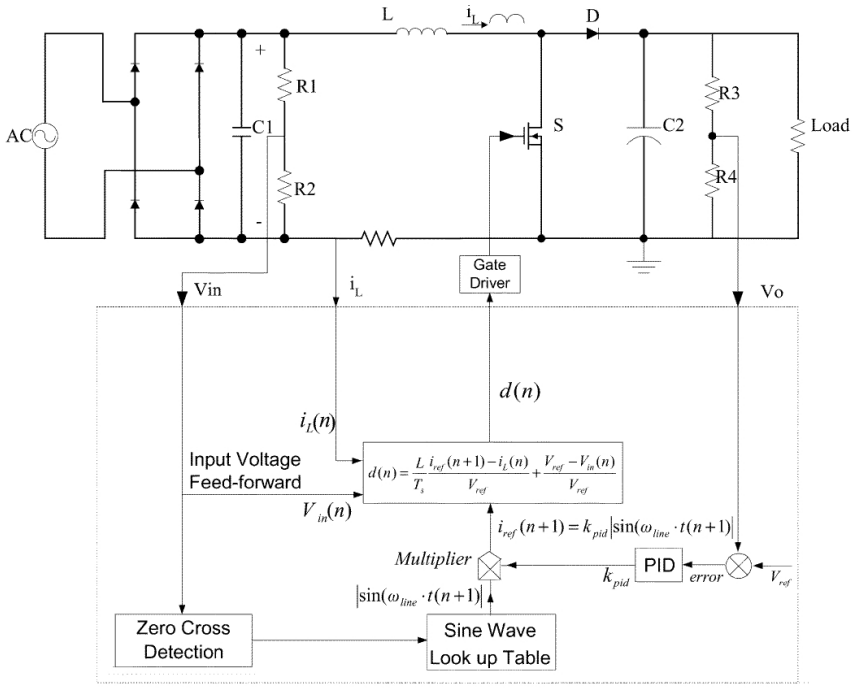


Figura 3.5: Sistema propuesto en [71].

A continuación se presenta en detalle la técnica para precalcular el ciclo de trabajo, así como diferentes formas de modificar los ciclos de trabajo precalculados en caso de que las condiciones de trabajo no sean las esperadas.

3.2. Precálculo del ciclo de trabajo

Las técnicas mostradas en este capítulo se basan en la aplicación de un conjunto de ciclos de trabajo aplicados al interruptor del corrector de factor de potencia. La conversión ac-dc es una tarea periódica con frecuencia igual al doble de la frecuencia de la red eléctrica, ya que la tensión alterna es rectificada. Gracias a su naturaleza periódica y determinista, únicamente hay que almacenar un conjunto de ciclos de trabajo, los cuales serán aplicados periódicamente. Aunque el cálculo de los ciclos de trabajo es dependiente de la topología del convertidor, en este capítulo se utilizará un convertidor elevador (figura 3.6) en modo de conducción continua°. Sin embargo, el cálculo de los ciclos de trabajo es similar para diferentes topologías.

La ecuación característica de una bobina en un convertidor elevador, dependiendo de si el interruptor está en conducción o en corte, es:

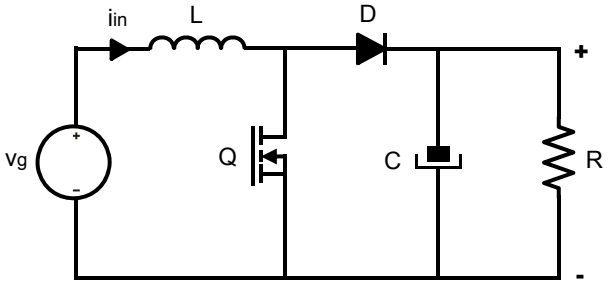


Figura 3.6: Topología de un convertidor elevador.

$$\begin{aligned} v_{L_{ON}} &= v_g = L \cdot \frac{d_{i_L}}{d_t} \\ v_{L_{OFF}} &= v_g - v_{out} = L \cdot \frac{d_{i_L}}{d_t} \end{aligned} \tag{3.3}$$

donde L es la inductancia de la bobina e i_L la corriente a través de la bobina. Por otra parte, v_g es la tensión de entrada y v_{out} es la tensión de salida. Dado que los ciclos de trabajo tienen naturaleza discreta debido a su implementación digital, estas ecuaciones se traducirán a ecuaciones en diferencias. Si se despeja la corriente que circula por la bobina, i_L, se obtiene:

$$\begin{aligned} i_L(k+1) = i_L(k) + \Delta i_{L_{ON}} + \Delta i_{L_{OFF}} = \\ i_L(k) + \frac{v_g(k)}{L} \cdot T_{Sw} \cdot d(k) + \frac{v_g(k) - v_{out}(k)}{L} \cdot T_{Sw} \cdot (1 - d(k)) \end{aligned} \tag{3.4}$$

Una vez discretizadas, el índice k indica el número de ciclo de conmutación dentro del conjunto de ciclos guardados en un semiciclo de red. Ademas, la constante T_{Sw} indica el periodo de conmutación, y d representa el ciclo de trabajo normalizado a la unidad. De esa forma, $T_{Sw} \cdot d(k)$ es el tiempo de carga de la bobina, mientras que $T_{Sw} \cdot (1 - d(k))$ es igual al tiempo de descarga.

Como se ha visto en la ecuación anterior, la corriente de entrada en cada ciclo depende del valor anterior de la corriente, de las tensiones de entrada y salida, y del ciclo de trabajo actual. Si se despeja el ciclo de trabajo en la ecuación (3.4), se obtiene:

$$d(k) = \frac{v_{out}(k) - v_g(k)}{v_{out}(k)} + \frac{L}{T_{Sw}} \cdot \frac{(i_L(k+1) - i_L(k))}{v_{out}(k)} \tag{3.5}$$

La ecuación (3.5) determina el ciclo de trabajo que tiene que aplicarse en el ciclo de conmutación k. Por tanto, la técnica de precálculo de ciclo de trabajo puede aplicarse almacenando previamente tantos valores como ciclos de conmutación haya en un semiperiodo de red. El ciclo de trabajo depende de la tensión de salida v_{out}. Como se vio en la figura 3.2, la tensión de salida tiene una componente de rizado debido a la corrección de factor de potencia. En particular, v_{out} es igual a:

$$v_{out_{Ripple}}(k) = \frac{P_{out}}{C \cdot 2 \cdot \omega_r \cdot V_{out}} \cdot \sin(2 \cdot \omega_r \cdot k \cdot T_{sw}) \tag{3.6}$$

Donde P_{out} es la potencia demandada por la carga, C la capacidad del condensador del filtro de salida. ω_r es la frecuencia angular de la red eléctrica, siendo igual a $2 \cdot \pi \cdot 50 \ \frac{rad}{s}$ ó $2 \cdot \pi \cdot 60 \ \frac{rad}{s}$, dependiendo del país donde se conecte el convertidor. Como resultado se puede ver que la frecuencia del rizado de la tensión de salida es el doble de la frecuencia de la red eléctrica, debido a la rectificación. Una vez definido su rizado, la tensión de salida se describe como:

$$v_{out}(k) = V_{out} - v_{out_{Ripple}}(k) \tag{3.7}$$

donde V_{out} es la componente continua de la tensión de salida.

Por otra parte, el ciclo de trabajo también depende de la corriente que circula por la bobina i_L. Dicha corriente depende de la potencia demandada por la carga y de la tensión de entrada:

$$i_L(k) = \frac{P_g}{V_g} \cdot \sqrt{2} \cdot \sin(\omega_r \cdot k \cdot T_{sw}) \tag{3.8}$$

P_g es la potencia media de entrada, que es igual a la potencia media de salida P_{out}, siempre que se ignoren las pérdidas eléctricas.

Una vez definida todas sus dependencias, la ecuación (3.5) determina el ciclo de trabajo que debe aplicarse en cada ciclo de conmutación. Es importante destacar que el precálculo de la ecuación (3.5) puede realizarse con toda la precisión deseada, ya que se puede calcular con un ordenador, mientras que el sistema final no tiene que realizar cálculos posteriores. Por tanto, se pueden añadir más fuentes de inexactitud en el cálculo de los ciclos de trabajo, tales como pérdidas eléctricas, sin hacer más complejo el sistema final, ya que éste únicamente recupera los valores almacenados.

En el caso ideal, la aplicación de los ciclos de trabajo obtendrá el factor de potencia óptimo para el convertidor diseñado. En realidad, diferentes factores como son las pérdidas, y las diferencias entre los parámetros de entrada teóricos y reales (potencia, tensión de entrada, conductancia, inductancia, frecuencia, etc), empeorarán el factor de potencia. Para paliar este empeoramiento, **es imprescindible añadir lazos cerrados de control**. En la siguiente sección se muestran diferentes métodos para añadir estos lazos de control.

3.3. Técnicas de PFC con ciclos de trabajo precalculados

Como se ha mencionado anteriormente, las técnicas de precálculo permiten simplificar los correctores de factor de potencia. El precálculo del ciclo de trabajo se realiza para unas condiciones determinadas, tales como tensiones de entrada y salida, potencia requerida, componentes usados en el convertidor, etc. El mayor problema de estas técnicas es que cualquier cambio en dichas condiciones provoca que el ciclo de trabajo calculado no sea el óptimo para el convertidor. La tensión de entrada y la tensión requerida de salida normalmente son menos variables, pero la potencia demandada sí suele fluctuar frecuentemente. Por otra parte, los componentes del convertidor, tales como la bobina, condensador, diodos, tienen ciertas tolerancias y derivas en sus parámetros característicos.

Estos problemas expuestos provocan tensiones de salida diferentes a la esperadas, y formas distorsionadas en la corriente de entrada, las cuales empeoran el factor de potencia. La solución es realizar una **regulación en tiempo real** durante el funcionamiento del convertidor para modificar los ciclos de trabajo precalculados. Sin embargo, la regulación de un conjunto de ciclos de trabajo no es una tarea trivial, como será visto durante esta sección. Además, se mostrarán diferentes técnicas de regulación sobre ciclos precalculados, ordenadas de menor a mayor complejidad.

El capítulo se va a centrar en métodos de regulación para corregir especialmente cambios en la tensión de entrada y, sobre todo, en la potencia de la carga. Es cierto

que los métodos mostrados serían también altamente dependientes de la frecuencia de la tensión de entrada. No se han desarrollado técnicas para actuar ante cambios de dicha frecuencia, asumiendo frecuencia constante a 50 Hz. Si el sistema debiera funcionar tanto a 50 Hz como a 60 Hz, se podrían precalcular sendos conjuntos de ciclos de trabajo. De esa forma, el sistema de sincronización con la red eléctrica (el cual se explicará en la sección 3.4) podría detectar fácilmente cuál de las dos posibles frecuencias está presente en la tensión de entrada, ya que la diferencia entre ellas es notable, y aplicaría el conjunto de ciclos de trabajo. Una vez detectada la frecuencia nominal de la tensión de entrada, es cierto que la tensión de entrada sufrirá con el tiempo pequeñas variaciones de frecuencia, las cuales afectarán a la corrección de factor de potencia. Estas variaciones pueden ser suficientemente grandes como para variar el número de ciclos de conmutación que se realizan en un semiciclo de red. En esta situación, si la frecuencia fuera menor de lo esperado, el sistema podría repetir ciclos de trabajo de forma equidistante durante el nuevo semiciclo de red. Si, en cambio, la frecuencia fuera mayor de lo esperado, el sistema eliminaría ciclos de trabajo, también de forma equidistante. La idea de repetir o eliminar de forma equidistante permite que la curva de ciclos de trabajo a aplicar no se vea distorsionada, haciendo que los ciclos de trabajo empiecen y acaben en 1 dentro de un semiciclo de red.

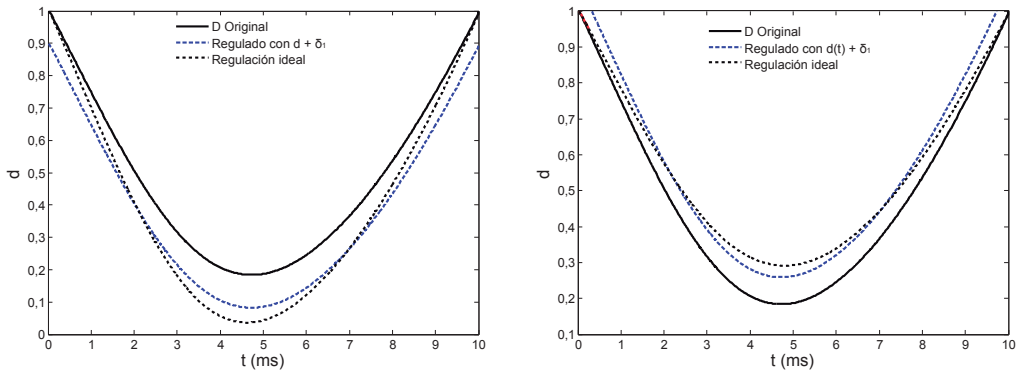
En cualquier caso, la frecuencia de la red eléctrica raramente se ve desviada más del 1 % de la nominal, por lo que dicho control no se ha considerado como crítico.

3.3.1. Regulación del ciclo de trabajo precalculado como un único componente

La idea más sencilla para regular el ciclo de trabajo es tener en cuenta la siguiente ecuación aplicable en modo de conducción continua:

$$< d >_{Tsc} = \frac{< v_{out} >_{Tsc} - < v_g >_{Tsc}}{< v_{out} >_{Tsc}} \tag{3.9}$$

Donde los valores $< d >_{Tsc}$, $< v_{out} >_{Tsc}$ y $< v_g >_{Tsc}$ están promediados durante un semiciclo de red. Si el valor promediado de la tensión de salida es diferente al esperado, todo el conjunto de ciclos de trabajo debe ser modificado proporcionalmente. Teniendo en cuenta esta ecuación, los cambios en la tensión de entrada o en la carga serán detectados midiendo la tensión de salida, y la tensión de salida podrá ser regulada cambiando el ciclo de trabajo. Además, si la consigna de la tensión de salida



(a) Regulación del $-15\,\%$.

(b) Regulación del $+15\,\%$.

Figura 3.7: Regulación sobre $d + \delta$.

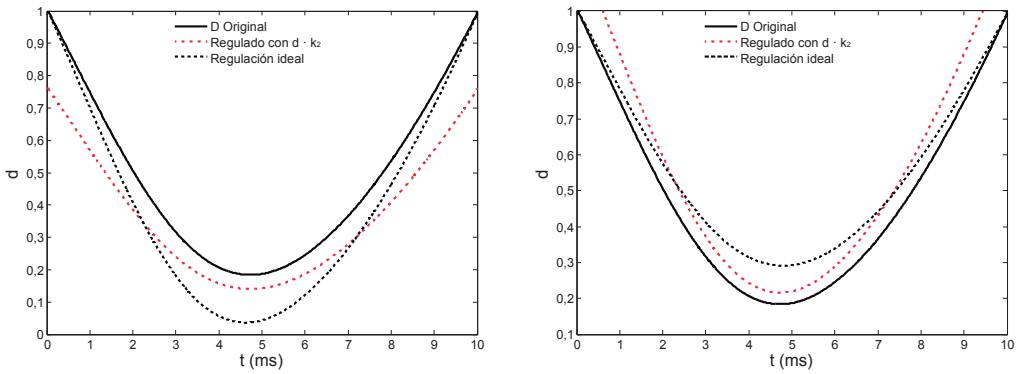
cambiara, igualmente se podría actuar cambiando el ciclo de trabajo. Aunque la regulación es muy sencilla, el problema que se presenta es que no sólo debe modificarse un ciclo de trabajo, sino el bloque entero precalculado.

Un método sencillo para cambiar cada ciclo de trabajo es **sumar o restar cierto valor a cada ciclo de trabajo** precalculado. De esa forma, el valor final del ciclo de trabajo en el ciclo de conmutación k, $d_{f1}(k)$ es:

$$d_{f1}(k) = d_o(k) + \delta_1 \tag{3.10}$$

donde $d_o(k)$ es el ciclo de trabajo precalculado durante el ciclo k, y δ_1 es la salida del regulador, la cual es constante durante un semiciclo de red. Esta regulación es válida para convertidores dc-dc, pero no es correcta en convertidores ac-dc, ya que distorsiona la curva de ciclos de trabajo, como se puede ver en la figura 3.7. El problema que se presenta es que el ciclo de trabajo en corrección de factor de potencia **debe empezar en** 1 para obtener factor de potencia alto. Si δ_1 fuera menor que 0, este método produce un resultado en el que los ciclos de trabajo de inicio y fin no empiezan en 1, como se ve en la figura 3.7(a). Por otra parte, si δ_1 fuera mayor que 0, este método satura el ciclo de trabajo a 1 durante varios ciclos de conmutación, lo cual también empeora el factor de potencia, como se puede ver en la figura 3.7(b).

Otro método es **multiplicar todos los ciclos de trabajo** durante el semiciclo de red por la salida del regulador, k_2:



(a) Regulación del $-15\,\%$.

(b) Regulación del $+15\,\%$.

Figura 3.8: Regulación sobre $d \cdot k_2$.

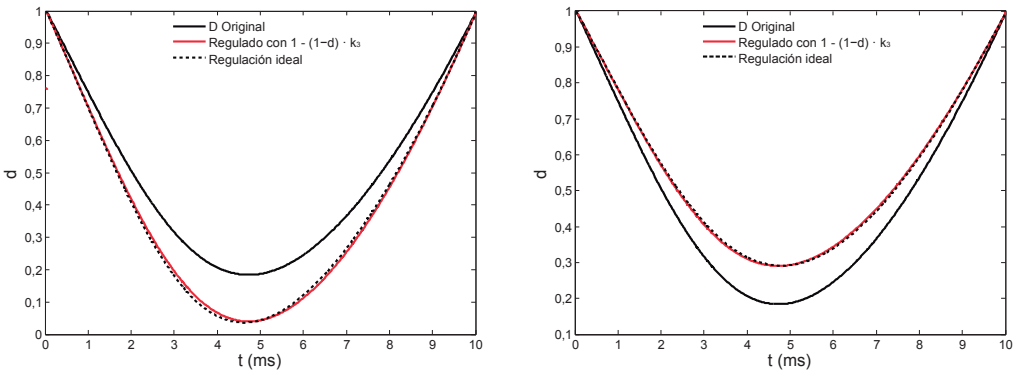
$$\begin{aligned} d_{f2}(k) &= d_o(k) \cdot k_2 \\ k_2 &= (1 + \delta_2) \end{aligned} \tag{3.11}$$

El nuevo conjunto, d_{f2}, es proporcional al ciclo original precalculado. La salida del regulador es igual a 1 en condiciones nominales, y se modifica alrededor de ese valor a través de cambios en δ_2. Al igual que en el método anterior, **la forma de inicio y fin de los ciclos de trabajo no se mantiene en** 1 cuando el regulador actúa. Este fenómeno se puede observar en la figura 3.8.

Para evitar la deformación vista en los dos métodos anteriores, se muestra otra forma de regulación. En vez de regular directamente el ciclo de trabajo d, **se puede regular el ciclo de trabajo usando** $(1 - d)$:

$$\begin{aligned} d_{f3}(k) &= 1 - (1 - d_o(k)) \cdot k_3 \\ k_3 &= (1 + \delta_3) \end{aligned} \tag{3.12}$$

Al igual que en la propuesta anterior, k_3 es la salida del regulador, siendo 1 en condiciones nominales, y modificándose alrededor de ese valor, a través de δ_3. Cuando $(1 - d)$ se multiplica por el valor del regulador, la curva de ciclos de trabajo **sigue empezando y acabando en** 1, dentro del semiciclo de red. Esto es debido a que los valores iniciales y finales del conjunto de ciclos de trabajo, los cuales están saturados a 1, son convertidos a 0 gracias a la operación $(1-d)$. Por tanto, cualquier multiplicación no modifica dichos valores iniciales y finales, una vez que la transformación se deshace.



(a) Regulación del $-15\,\%$.

(b) Regulación del $+15\,\%$.

Figura 3.9: Regulación sobre $1-(1-d)\cdot k_3$.

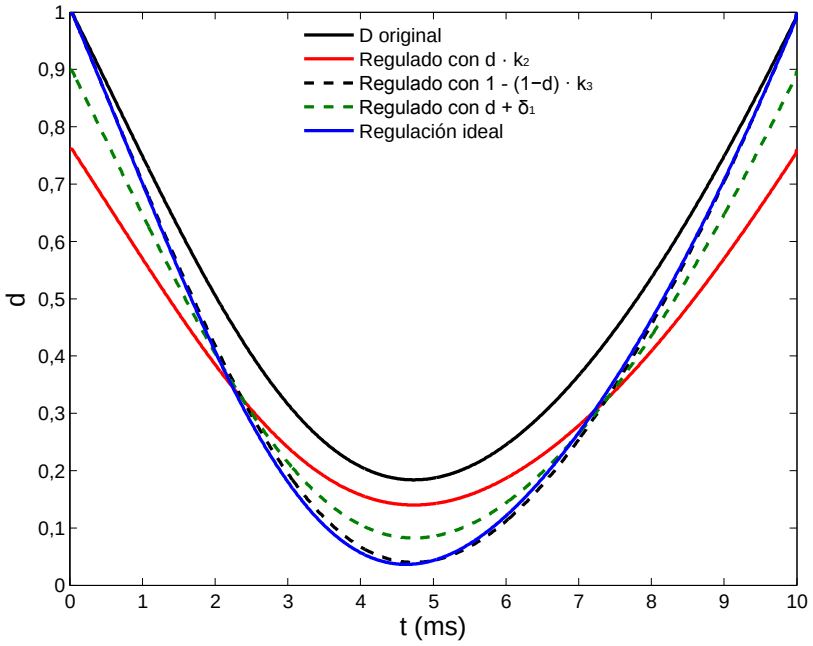
La regulación de $(1-d)$, en vez de regular directamente d, es una aportación original de esta tesis.

La figura 3.9 muestra cómo se modifica el ciclo de trabajo con este último método propuesto. Como se puede observar, la regulación obtenida con este método es muy parecida a la ideal, siendo mucho más precisa que las dos propuestas anteriores.

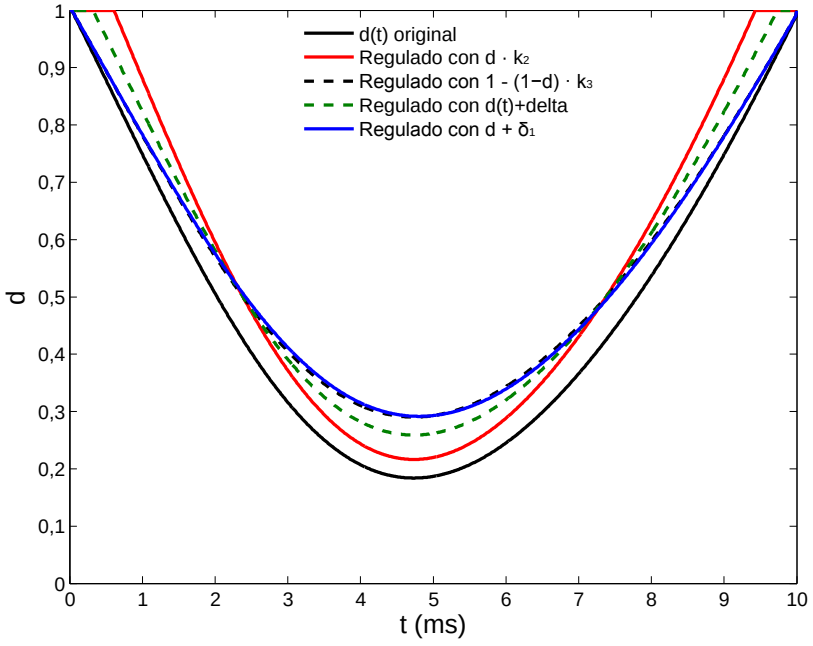
Por último, las tres propuestas están resumidas en la figura 3.10. La figura muestra claramente el **mejor resultado del tercer método propuesto, usando $1-d$**. Se han tomado resultados experimentales para comparar los tres métodos en la sección 3.6.1. Los resultados demuestran nuevamente que el tercer método obtiene mejores resultados. Debido a ello, a partir de ahora se tomará en cuenta el tercer método cuando se hable de regulación sobre una única componente del ciclo de trabajo.

Una vez descrita la regulación óptima para este método, se puede definir el regulador. Éste se puede hacer con un regulador PID sencillo, el cual actuará sobre la ecuación (3.5). De esta forma, el regulador se comporta de forma **similar a un lazo de tensión en un corrector clásico de factor de potencia**. El regulador simplemente mide la tensión media de salida, y cambia los ciclos de trabajo de acuerdo a la medida.

El lazo de tensión propuesto con esta técnica se muestra en la figura 3.11. Como puede observarse, la salida del regulador, k o k_3, se multiplica con el ciclo de trabajo complementario $(1-d)$, obteniendo el ciclo regulado complementario $(1-d)^*$. Finalmente, este ciclo complementario se traduce a ciclo de trabajo, d^* o d_{f3}. El regulador se muestra con mayor detalle en la figura 3.12. Como se comentó anteriormente, el regulador se basa en un controlador PID y su salida, δ, es sumada a 1, por lo que la salida final es $1+\delta$. En condiciones nominales, δ es igual a 0, por lo que la salida k es



(a) Regulación del $-15\,\%$.



(b) Regulación del $+15\,\%$.

Figura 3.10: Regulación sobre los tres métodos propuestos para componente única en el ciclo de trabajo.

1. Sin embargo, δ, y por tanto k, son incrementados o decrementados alrededor del punto de equilibrio para controlar la tensión de salida, según ordene el controlador PID.

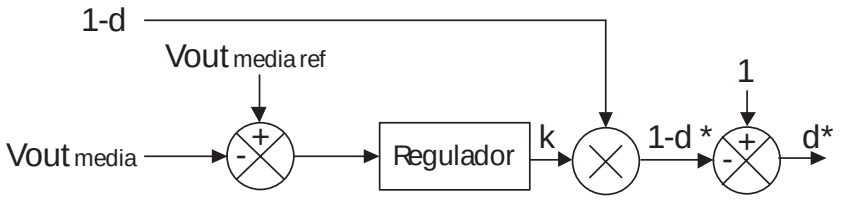


Figura 3.11: Sistema de control usando d como un único componente.

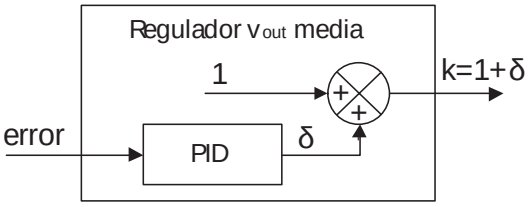


Figura 3.12: Regulador usado para controlar d con el lazo de tensión medida de salida.

En definitiva, este método propuesto usa la ecuación (3.5), la cual se calcula para condiciones nominales. Gracias al lazo descrito, la tensión de salida es regulada, haciendo más robusto el corrector de factor de potencia. Sin embargo, **los cambios en la carga no son bien detectados por este lazo**, ya que los cambios en la carga no producen cambios significativos en la tensión media de salida, siempre que la eficiencia del convertidor sea alta. Por tanto, el lazo no detecta correctamente los cambios de carga, no se ajustará el ciclo de trabajo adecuadamente, y el factor de potencia decrecerá. Esta limitación será tratada con los métodos que se describen a continuación.

3.3.2. Regulación del ciclo de trabajo precalculado como dos componentes

El anterior método descrito no es capaz de controlar correctamente los cambios en la carga del convertidor. A continuación se analiza el ciclo de trabajo con más detalle y se mostrará cómo detectar cambios en la carga.

El ciclo de trabajo descrito en la ecuación (3.5) puede dividirse en d_1 y d_2:

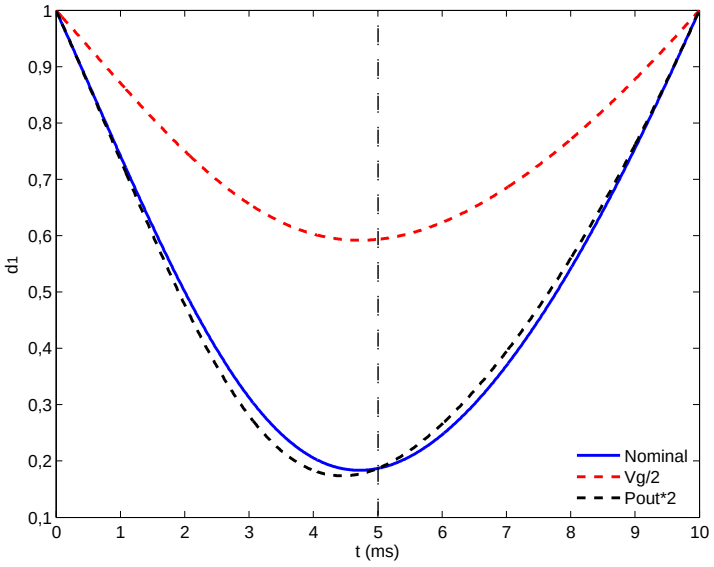
$$\begin{aligned} d_1(k) &= \frac{v_{out}(k) - v_g(k)}{v_{out}(k)} \\ d_2(k) &= \frac{L}{T_{Sw}} \cdot \frac{(i_L(k+1) - i_L(k))}{v_{out}(k)} \\ d(k) &= d_1(k) + d_2(k) \end{aligned} \tag{3.13}$$

Los parámetros d_1 y d_2 se muestran en las figuras 3.13(a) y 3.13(b) respectivamente. Como puede verse, d_1 **es la componente principal** del ciclo de trabajo mientras que, como se verá, d_2 **permite corregir la distorsión en la corriente de entrada producida por la carga**. La figura 3.13(a) muestra que d_1 es fuertemente influenciada por las variaciones de la tensión de entrada y en menor medida por la potencia del convertidor. Teniendo en cuenta la ecuación previa, se pueden observar estas dependencias. La dependencia con la tensión de entrada es obvia, y la dependencia con la potencia viene dada por la componente de rizado de v_{out}, la cual depende de la potencia del convertidor, como se vio en la ecuación (3.6). La relación entre potencia y la componente d_1 es visible en la figura 3.13(a), la cual muestra cómo d_1 no es simétrica. Esta asimetría es provocada precisamente por el efecto de la carga, creciendo cuanto mayor sea la potencia demandada. Dado que la dependencia con la tensión de entrada es mayor comparada con la carga, la componente d_1 puede ser controlada únicamente midiendo la tensión media de salida. Es cierto que se ignora esa menor dependencia de la potencia, pero el error cometido no es, a priori, excesivo.

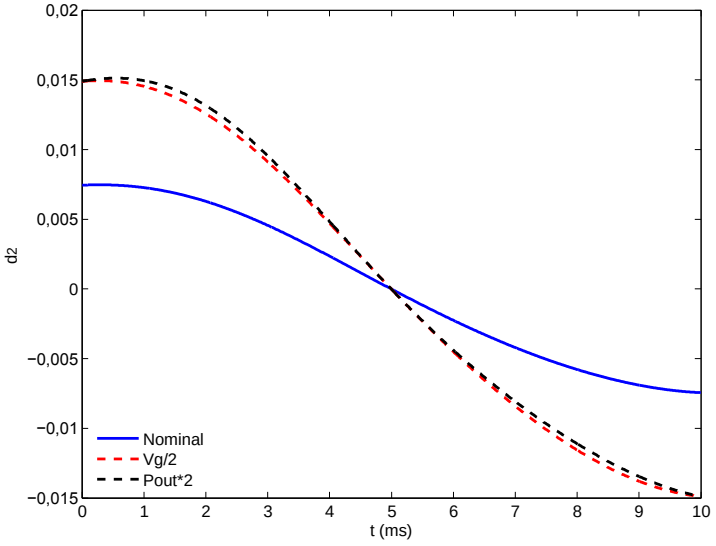
El regulador de d_1 es similar al descrito en el método anterior, midiendo la tensión media de salida, y actuando como un lazo clásico de tensión. De forma similar, lo ideal es guardar $(1 - d_1)$ en vez de d_1, para así no distorsionar el ciclo de trabajo cuando el regulador lo modifique.

Por su parte, la componente d_2 depende de la corriente de entrada y de la tensión de entrada, como muestra la figura 3.13(b). La corriente de entrada, como puede verse en la ecuación (3.8), es proporcional a la potencia del convertidor. Por tanto, cualquier cambio en la potencia demandada por la carga afectará proporcionalmente a la corriente de entrada, y la componente d_2 debe modificarse. La corriente de entrada podría medirse mediante un sensor de corriente, como en un corrector de factor de potencia clásico. Sin embargo, la idea de aplicar ciclo precalculado precisamente busca el ahorro del número de sensores, y especialmente del sensor de corriente. Dado que el sistema ya tiene un conversor analógico digital para medir la tensión de salida (lazo de tensión para regular d_1), este mismo sensor puede utilizarse para medir indirectamente la corriente de entrada. Esto es posible ya que el rizado de la tensión de salida también es proporcional a la potencia del convertidor, como muestra la ecuación (3.6). De esta forma, **el rizado en la tensión de salida es proporcional a la potencia y, por tanto, también a la componente** d_2.

Si cambia la tensión de entrada, para la misma potencia cambiará también la corriente de entrada, pero en sentido inverso. De esa forma, d_2 también se ve influenciada por la tensión de entrada y, debido a ello, dicha componente está regulada



(a) d_1.



(b) d_2.

Figura 3.13: Formas de las componente d_1 y d_2 durante un semiciclo de red.

nuevamente mediante el lazo de tensión media de salida.

El flujo de control para el método que se está describiendo se muestra en la figura 3.15. Como se puede observar, hay un lazo de tensión media de salida para regular d_1, siendo similar al lazo de tensión en un corrector de factor de potencia clásico. Además, la componente d_2 **es regulada por dos lazos**, el lazo de tensión media de salida, pero también el lazo del rizado de la tensión de salida. Este último **se comporta como un lazo de corriente en un corrector clásico**, pero la diferencia principal

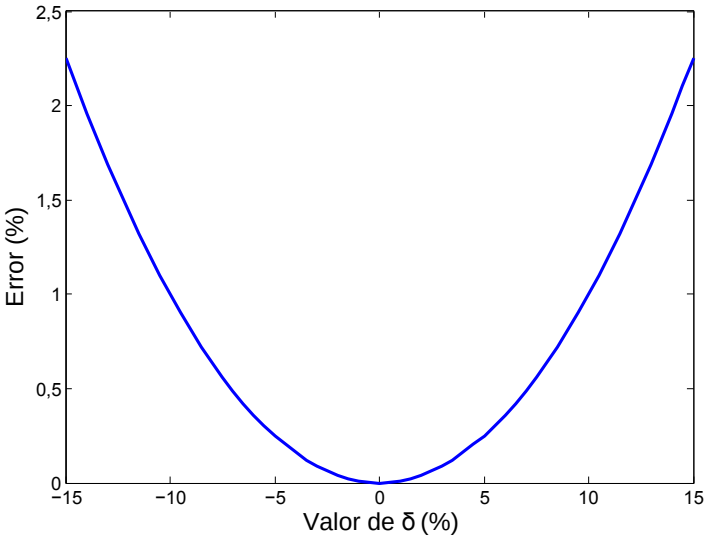


Figura 3.14: Error producido al usar $1-\delta$ en vez de $\frac{1}{1+\delta}$.

es que usa el rizado de la tensión de salida, en vez de la corriente de entrada. Este lazo, además, tiene un ancho de banda notablemente menor que un lazo de corriente clásico, ya que sólo actúa una vez por semiciclo de red. Debido a ello, el ADC necesario para el lazo puede tener baja frecuencia de muestreo y ser de bajo coste.

De forma similar al método anterior, presentado en la sección 3.3.1, la primera componente $(1-d_1)$ se regula usando $k=1+\delta$. Sin embargo, la componente d_2 se almacena directamente, no usando el término complementario $(1-d_2)$. Por tanto, para que el lazo de tensión media regule d_2, la salida del lazo debe ser $\frac{1}{k}$ en vez de k, ya que los términos $(1-d_1)$ y d_2 tienen signos opuestos. El problema principal es que la división $\frac{1}{k}$ requiere recursos *hardware* no despreciables. Una forma de aproximar el resultado de la división es suponer que $\frac{1}{k}=\frac{1}{1+\delta}$ **es similar a** $\frac{1}{k}''=1-\delta$, ya que δ oscila alrededor de 0. Esta aproximación produce un error, pero como se verá con el siguiente ejemplo, es bastante pequeño. Como ejemplo, si la tensión de entrada es 10 % menor de lo esperado, δ será igual a $-0{,}1$, por lo que $\frac{1}{k}$ es aproximadamente 1,111 y $\frac{1}{k}''=1{,}100$. En este caso, un error del 10 % en la tensión de entrada produce un error del 1,01 % en la regulación de la componente d_2. Además, es importante destacar que la diferencia con la tensión de entrada esperada suele ser bastante menor al 10 %. La figura 3.14 muestra el error cometido al realizar esta simplificación. Como se puede ver, las regulaciones por debajo del 5 %, lo cual es razonable para el lazo de tensión, producen errores por debajo del 0,25 % al realizar la simplificación comentada.

La aproximación descrita **reduce drásticamente los recursos *hardware* necesarios**, ya que una división es mucho más costosa en términos de área y procesamiento

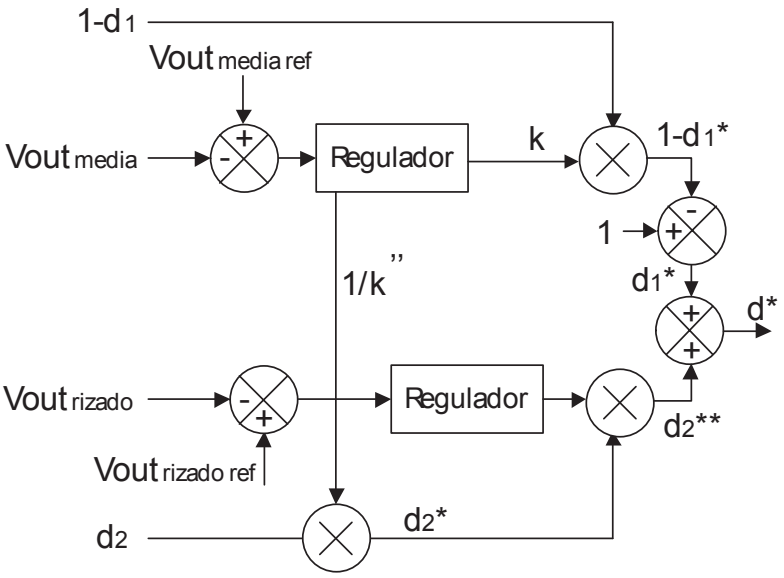


Figura 3.15: Sistema de control usando las componentes d_1 y d_2.

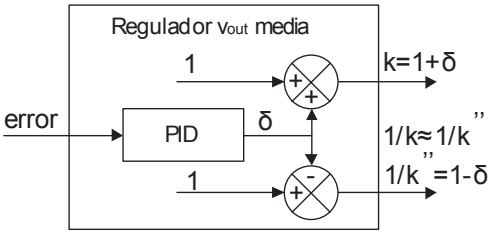


Figura 3.16: Regulador usado para controlar las componentes d_1 and d_2 utilizando el lazo de tensión media de salida.

que un sumador. La figura 3.16 muestra el regulador comentado, el cual tiene dos salidas, k para regular $(1 - d_1)$, y $\frac{1}{k}''$ para regular d_2.

La división del ciclo de trabajo en dos parámetros, d_1 y d_2, fue presentada por otros autores en [70, 71]. Sin embargo, en [70], se presentó un algoritmo predictivo, en el que el conjunto de ciclos de trabajo para el próximo semiciclo de red se calculan en el semiciclo actual. Además, en su algoritmo se mide la tensión de entrada, incrementando el coste del sistema. Por otra parte, en [71], los mismos autores proponen una mejora al sistema añadiendo el sensado de la corriente de entrada, aumentando la robustez del sistema, pero sin reducir el número de sensores en el corrector. En contraste, en este capítulo se muestran técnicas para reducir el número de medidas, consiguiendo medir únicamente la tensión de salida. Además, se propone en la siguiente sección otra separación más fina del ciclo de trabajo en tres componentes.

3.3.3. Regulación del ciclo de trabajo precalculado como tres componentes

En el anterior método se comentó que la componente d_1 estaba fuertemente influenciada por la tensión de entrada, y ligeramente por la potencia del convertidor (figura 3.13(a)). Debido a ello, la regulación de d_1 usando únicamente el lazo de tensión de salida media puede ser suficiente. En el método que se va a describir se añade, además, la regulación requerida para que d_1 se modifique en caso de cargas no nominales. De esta forma el factor de potencia se verá alterado en menor medida cuando la potencia demandada no sea la nominal.

El método propuesto se basa en dividir la componente d_1 en dos componentes d_a y d_b:

$$\begin{aligned} d_1(k) &= \frac{v_{out}(k) - v_g(k)}{v_{out}(k)} \\ d_a(k) &= \frac{V_{out} - v_g(k)}{V_{out}} \\ d_b(k) &= d_1(k) - d_a(k) \end{aligned} \tag{3.14}$$

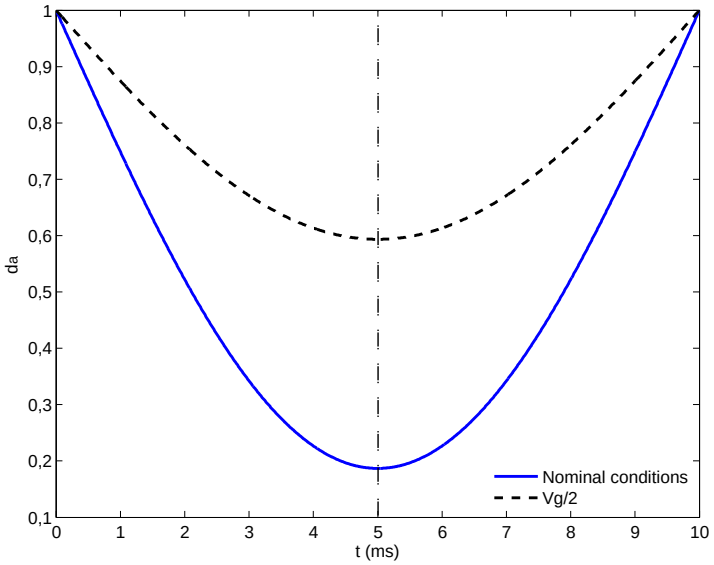
Las formas de onda de d_a y d_b se muestran en las figuras 3.17(a) y 3.17(b) respectivamente. El parámetro d_a **define la relación entre tensiones de entrada y salida**, similar a la ecuación (3.9). Por tanto, d_a no depende de la potencia de la carga, siendo simétrica. d_b **es el resultado de restar** d_a **a** d_1**, por lo que depende tanto de la tensión de entrada como de la carga**.

Además, la componente anteriormente llamada d_2 se va a renombrar a d_c para mantener coherencia en la nomenclatura:

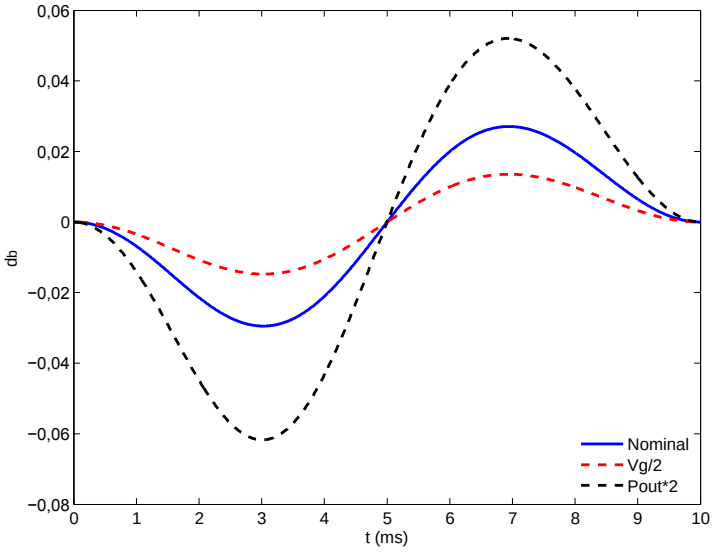
$$d_c(k) = d_2(k) = \frac{L}{T_{sw}} \cdot \frac{(i_L(k+1) - i_L(k))}{v_{out}(k)} \tag{3.15}$$

Por último, el ciclo de trabajo d se calcula sumando las componentes d_a, d_b y d_c:

$$d(k) = d_a(k) + d_b(k) + d_c(k) \tag{3.16}$$



(a) d_a.



(b) d_b.

Figura 3.17: Formas de las componente d_a y d_b durante un semiciclo de red.

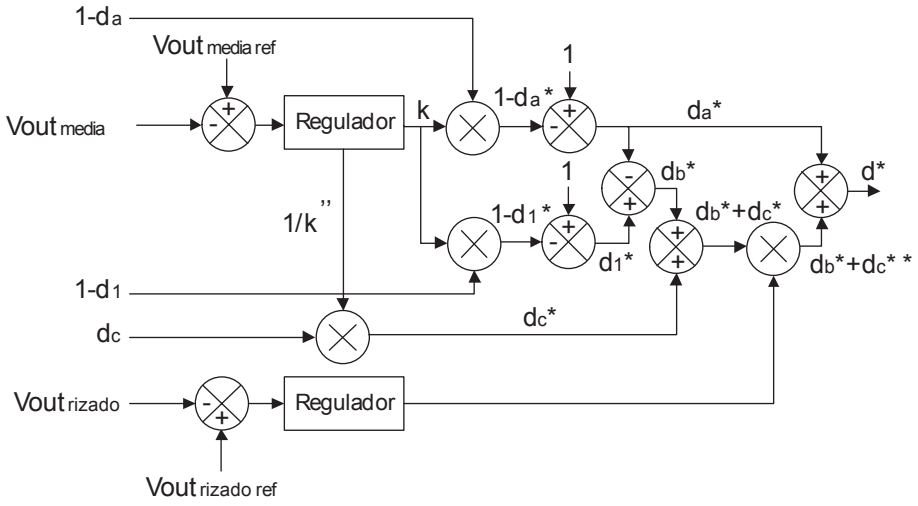


Figura 3.18: Sistema de control usando d_a, d_b y d_c.

Según muestra la ecuación (3.15), los valores de d_a y d_1 deben calcularse para obtener d_b. Para ser robusto ante cambios de tensión de entrada, d_a y d_1 deben ser controlados con el lazo de tensión media de salida. Además, como novedad en este método, el parámetro d_b, una vez calculado en cada ciclo de conmutación, debe ser regulado con el lazo de rizado de la tensión de salida. Esto es debido a que d_b es dependiente de la potencia, como se ha comentado. Por último y como se ha comentado, d_c es igual a d_2, por lo que se sigue regulando con ambos lazos, al igual que en el anterior método.

La arquitectura del regulador propuesto se muestra en la figura 3.18. Como puede observarse, el regulador de tensión media de salida se usa tres veces para controlar las componentes $(1 - d_a)$, $(1 - d_1)$ y d_c. Por su parte, el regulador del rizado de la tensión de salida se usa una vez para obtener $(d_b^* + d_c^*)^*$, es decir, para regular la componente $(d_b^* + d_c^*)$, la cual ya ha sido regulada por el lazo de tensión media de salida. Esta división en tres componentes es una **aportación original de este capítulo de tesis**.

3.4. Sincronización con la red eléctrica

La técnica de ciclo precalculado se basa en la idea de aplicar un conjunto de ciclos de trabajo calculados con anterioridad. Cada ciclo de trabajo dentro del conjunto está calculado para aplicarse en un momento preciso dentro del semiciclo de red. Por tanto, es crítica la sincronización entre la memoria de ciclos de trabajo y la red eléctrica.

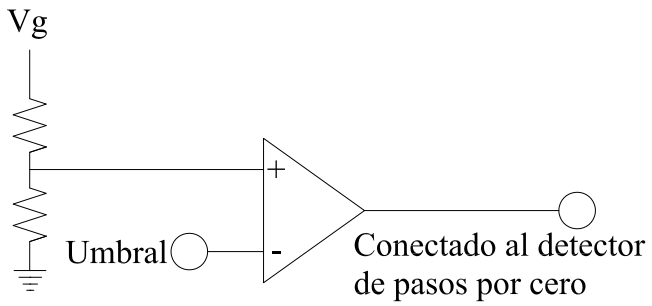


Figura 3.19: Circuito necesario para detectar el paso por cero de la tensión de entrada.

Una opción fácil sería añadir un ADC para medir la tensión de entrada y empezar a aplicar los valores guardados cuando dicha tensión se aproxime a 0 V. Sin embargo, y como ya se ha comentado, se quiere reducir el número de ADCs para el sistema propuesto.

Otra posibilidad es **usar un comparador de tensión**, el cual cambiará su salida, 1 por ejemplo, cuando la tensión de entrada supere un umbral establecido, como muestra la figura 3.19. El umbral podría estar muy cercano a 0 V, pero la medida alrededor del paso por cero es muy ruidosa debido a la rectificación. Para alejarse de la zona ruidosa, se puede aumentar el umbral, haciendo que el comparador fije un 1 en su salida durante un tiempo mayor. En general, el paso por cero se producirá aproximadamente en el instante que coincide con la mitad del tiempo en el que el comparador tiene su salida a 1, siempre que el comparador no conmute repetitivamente en torno al punto de umbral. Analizando el tiempo a 1 del comparador, se puede calcular a posteriori el instante en el cual el comparador llevaba la mitad de tiempo a 1. Dado que el comportamiento entre semiciclos de red es prácticamente idéntico, se puede usar ese cálculo para predecir cuándo se producirá el nuevo paso por cero, en el siguiente semiciclo de red.

Dado que la salida del comparador será ruidosa en torno al umbral de tensión, **es necesario realizar un filtrado**.

La figura 3.20 muestra un esquema de la implementación realizada para la sincronización con la red eléctrica. Se puede observar que hay un contador el cual en cada ciclo de reloj se incrementa o decrementa en uno. Su valor se incrementará cuando el comparador esté a 1, mientras que se decrementará cuando el comparador tenga su salida a 0. El valor mínimo del contador es 0, saturándose a ese valor en caso necesario, mientras que el valor máximo dependerá de la frecuencia de la red, y de la tensión de entrada. Un registro guarda el valor máximo del contador alcanzado durante el presente semiciclo de red. A partir de cierto instante de tiempo, se puede calcular el valor medio del contador, dividiendo entre dos el valor máximo del

contador. Esto puede ser realizado fácilmente desplazando un bit hacia la derecha el registro del valor máximo. El momento de cálculo puede estar predeterminado, por ejemplo, cuando haya transcurrido la mitad del tiempo equivalente a un semiciclo de red. Otra opción es contar un tiempo a partir del último máximo adquirido, y tras él, calcular el valor medio. En cualquier caso, es valor medio del contador se usará como consigna para detectar el próximo paso por cero.

Como se comentó, el contador anterior decrece cuando el comparador establece un 0 en su salida. Esta funcionalidad permite filtrar el ruido que se genera cuando la tensión de entrada está cercana al umbral de comparación. Este filtro sencillo permite calcular de forma bastante fiable el instante donde se ha producido el paso por cero, para así predecir el siguiente paso por cero. Cuando se acerque el próximo paso por cero, el contador volverá a establecer su salida a 1, haciendo crecer de nuevo el contador. Llegará un momento en el que el contador cruce la consigna generada en el anterior semiciclo de red. En ese preciso momento, se genera una señal de sincronismo, que servirá para reiniciar el contador de direcciones de las memorias que almacenan los ciclos precalculados.

Este proceso se produce durante todos los semiciclos de red mientras que el convertidor esté funcionando. En general, la consiga se mantiene estable, ya que la frecuencia de la red es fija, pero ésta podría variar muy ligeramente. Además, debido al ruido, y a pesar del filtro descrito, puede haber pequeñas variaciones en la consiga. Por ello, se vuelve a filtrar la señal de la consigna, guardando las últimas consignas, y realizando la media. De nuevo, se puede realizar la media de forma trivial siempre que el número de muestras sea potencia de dos. En particular, en el sistema propuesto se toman las últimas cuatro medidas y se desplaza el sumatorio dos posiciones hacia la derecha, equivalente a dividir entre cuatro.

3.5. Análisis de resolución y efectos de la cuantización

En los convertidores de potencia controlados digitalmente, es importante analizar la resolución del ADC y los efectos de cuantización que se producen. En particular, si no se tienen en cuenta la resolución del ADC (sensor) y del actuador (normalmente un PWM digital), se puede producir el **efecto del ciclo límite en el lazo de tensión**. El ciclo límite ha sido ampliamente descrito en la literatura. Por ejemplo, [30, 29] analizan el ciclo límite en convertidores dc-dc. En este tipo de convertidores, el ciclo límite se produce cuando la resolución del PWM es menor que la resolución del ADC. Si esa condición se cumple, es posible que el regulador no encuentre un punto con error nulo, y se produzcan oscilaciones de baja frecuencia, fenómeno también

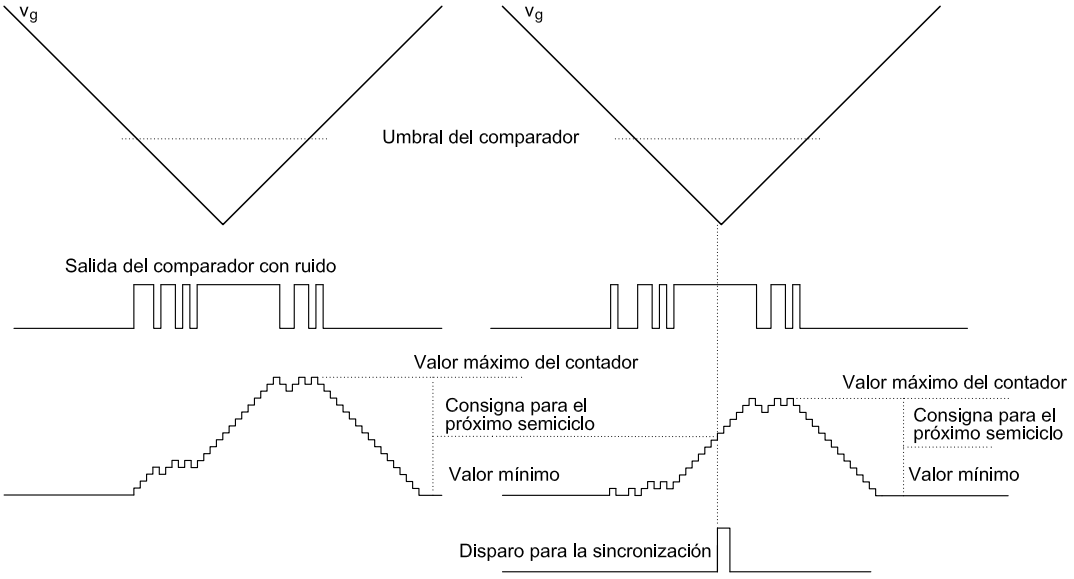
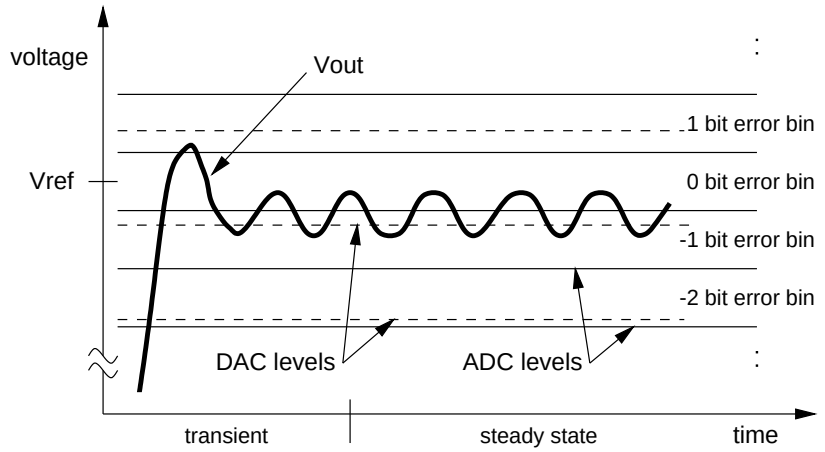


Figura 3.20: Sincronización con la red eléctrica.

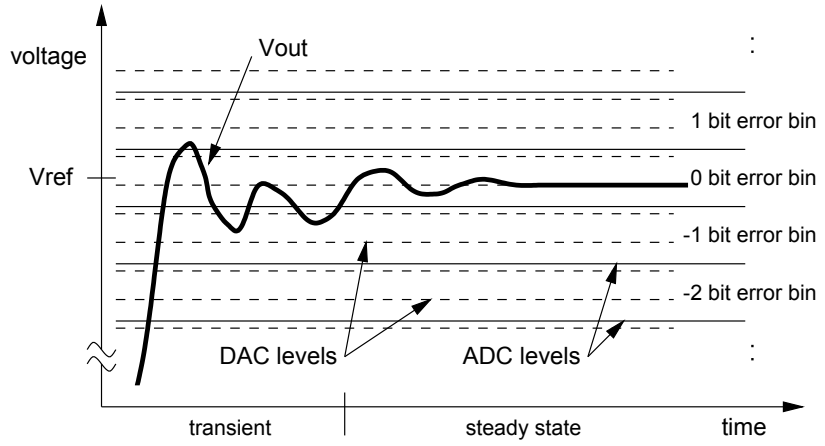
llamado ciclo límite. La figura 3.21(a) muestra un caso en el que hay ciclo límite. En la figura, las líneas horizontales punteadas delimitan los posibles puntos de operación del actuador, en su equivalencia a tensión de salida. Por otra parte, las líneas horizontales no punteadas separan las zonas que el ADC mide, y dentro de una zona el ADC no puede distinguir diferentes valores de tensión. De hecho, las líneas estarán más juntas según la resolución sea mayor. En la figura 3.21(a) hay ciclo límite ya que no hay un punto de actuación que resida en la zona de error nulo del ADC. Por otra parte, la figura 3.21(b) muestra un caso en el que no hay ciclo límite, ya que al menos hay un punto de actuación que se encuentra en una zona de error nulo del ADC.

El análisis de la resolución y de los efectos de cuantización es más complejo en el caso de la corrección de factor de potencia. En [72] se realiza el citado análisis para PFC con técnicas clásicas. Dichas técnicas tienen un lazo de tensión que genera un comando de potencia, G_{in} o conductancia de entrada, el cual se usa en el lazo de corriente. En [72] se asume que el lazo de corriente es perfecto, por lo que éste no afecta a la tensión media de salida. De esa forma, sólo habría que ver los efectos de cuantización para el lazo de tensión.

Las técnicas de ciclo precalculado que han sido propuestas no tienen dos lazos en serie, sino que el ciclo de trabajo se aplica directamente al interruptor, salvo la regulación previa que haya. La propuesta que obtiene mejores resultados, presentada en la sección 3.3.3, tiene dos lazos de control, aunque diferentes a los usados en la corrección de factor de potencia tradicional. El mejor método propuesto divide el ciclo de trabajo en tres componentes: d_a, d_b y d_c. Los parámetros d_b y d_c no modifican la



(a) Hay ciclo límite dado que la resolución de la actuación es menor que la del sensor.



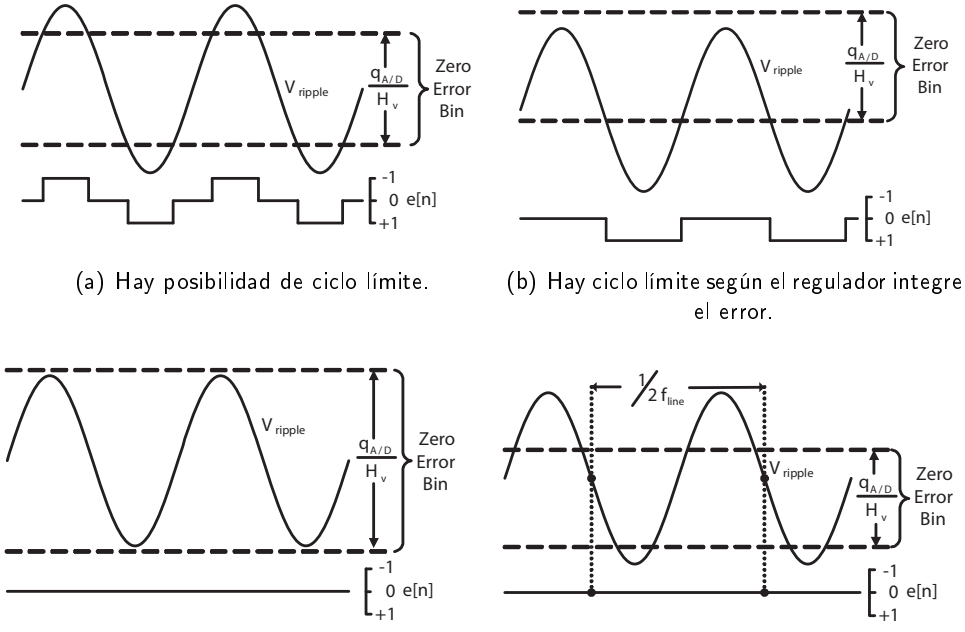
(b) La resolución de la actuación es mayor que la del sensor y, por tanto, no hay ciclo límite.

Figura 3.21: Ciclo límite en convertidores dc-dc. Imágenes extraídas de [30].

tensión media de salida, puesto que su valor medio durante un semiciclo de red es nulo (figuras 3.17(b) y 3.13(b)). **El único parámetro que afecta a la tensión media de salida es** d_a, mostrado en la figura 3.17(a). El análisis de resolución, por tanto, se realizará teniendo en cuenta únicamente el parámetro d_a.

Basándose en el análisis detallado en [72], se estudiarán las condiciones que han de cumplirse para evitar la aparición del ciclo límite. El ciclo límite no es deseable ya que produce oscilaciones subarmónicas en la tensión de salida y en la corriente de entrada. Hay dos grandes fuentes de generación de ciclo límite.

La primera fuente de ciclo límite en convertidores ac-dc es el muestreo de la tensión de salida, ver figura 3.22. La figura 3.22(a) muestra un caso donde seguramente se producirá ciclo límite. En la figura se ve cómo la tensión de salida del convertidor



(a) Hay posibilidad de ciclo límite.

(b) Hay ciclo límite según el regulador integre el error.

(c) No hay ciclo límite ya que el rizado está dentro de la ventana del ADC.

(d) No hay ciclo límite ya que la medida está sincronizada con el rizado.

Figura 3.22: Ciclo límite en PFC según la sincronía de las medidas. Imágenes extraídas de [72].

oscila entre zonas de error de ± 1 LSB. La única posibilidad de que no haya ciclo límite es que el número de veces que el error es -1 sea igual al número de veces que el error es $+1$. Por tanto, es bastante razonable pensar que esta situación conducirá a situaciones de ciclo límite. La figura 3.22(b) muestra un caso donde hay seguridad en la existencia de ciclo límite. En ella se muestra que la tensión de salida oscila entre dos zonas, con error nulo y error siempre positivo. Por tanto, el regulador irá integrando el error cometido y cambiará su actuación.

Por otra parte, la figura 3.22(c) ilustra un ejemplo donde no hay ciclo límite, dado que la ventana del ADC es suficientemente grande para abarcar el rizado de la tensión de salida. El error siempre será 0 y no habrá ciclo límite, pero este método sacrifica resolución en el ADC, no siendo óptima. Por último, la figura 3.22(d) muestra un caso en el que no existe ciclo límite. En este caso, aunque la tensión de entrada cruce diferentes zonas de error, la medida es tomada siempre en el mismo punto dentro de un semiciclo de red. Por tanto, no habrá ciclo límite, ya que el regulador puede llegar a una actuación en la que dicho punto se mantenga en una zona de error nulo.

En el caso propuesto, la tensión de salida se muestrea múltiples veces y se realiza una media de todos los valores obtenidos. El sistema estará libre de ciclo límite producido por el muestro porque el comando de potencia (salida del regulador de

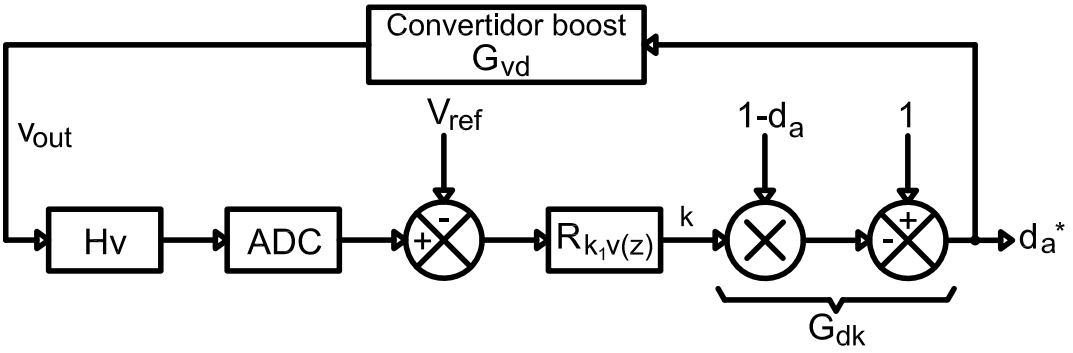


Figura 3.23: Modelo del lazo de tensión media de salida para su análisis de resolución y cuantización.

tensión) sólo se produce una vez por semiciclo de red y además se produce en el mismo punto, específicamente en el paso por cero de la tensión de entrada.

La segunda fuente de generación de ciclo límite se puede derivar de la cuantización del comando de potencia. Normalmente el comando de potencia es la salida del lazo de tensión, la cual sirve de entrada al lazo de corriente. En cambio, en la propuesta de este capítulo, el comando de potencia equivale al valor k del regulador de tensión, que sirve para cambiar la componente d_a, como se explicó en la sección 3.3.3.

El ciclo límite a causa del comando de potencia puede surgir tanto en régimen permanente como durante transitorios. En el primer caso, ocurrirá cuando el comando no tiene suficiente resolución para obtener error nulo en la medida del ADC, mientras que en el segundo ocurrirá cuando la acción integral del regulador es demasiado grande. La figura 3.23 muestra un modelo del lazo de tensión media de salida respecto al comando de potencia k. En la figura, H_v es la ganancia del circuito de acondicionamiento de la tensión de salida, el cual se utiliza previo a la medición con el ADC. Normalmente este circuito es un divisor resistivo. ADC es el convertidor analógico-digital, y $Rk_1v(z)$ es el regulador, el cual tiene en cuenta el valor medio de la tensión de salida para generar k_1. Por último el valor de k_1 se traducirá a un cambio del valor de d_a, como se vio en la figura 3.18, y a través de la ecuación:

$$D_a^* = 1 - (1 - d_a) \cdot k_1 \tag{3.17}$$

La condición para evitar el ciclo límite en régimen permanente es que el paso mínimo que puede realizar el regulador sea menor que el paso mínimo del ADC. De esa forma, siempre habrá una solución en la que el error (referencia - medida) de la

tensión media de salida sea nulo, (ver figura 3.21(b)). Esta condición se ve reflejada en:

$$G_{vk0} \cdot H_v \cdot q_k < q_{ADC} \tag{3.18}$$

donde G_{vk0} es la ganancia en modelo de pequeña señal y en continua del comando de potencia k. q_k y q_{ADC} son las resoluciones del regulador y del ADC respectivamente, es decir, el valor de sus bits menos significativos. Si la ecuación (3.18) se cumple, siempre habrá un valor del comando de potencia que resida dentro de la zona de error nulo del ADC. La ganancia del comando de potencia, G_{vk0}, puede ser dividida en:

$$G_{vk0} = G_{vd0} \cdot G_{dk0} \tag{3.19}$$

G_{dk0} es la ganancia en modelo de pequeña señal y en continua desde el comando de potencia k hasta el ciclo de trabajo d_a, y G_{vd0} es la ganancia en modelo de pequeña señal y en continua desde el ciclo de trabajo hasta la tensión de salida. En la tercer método propuesto, k se multiplica por $(1 - d_a)$ y después se resta a 1, por lo que G_{dk0} es igual a $(1- < d_a >_{Tu})$, donde $< d_a >_{Tu}$ es el valor medio de d_a durante un semiciclo de red. Por otra parte, G_{vd0} es la ganancia en modelo de pequeña señal y en continua del convertidor elevador desde el ciclo de trabajo hasta la tensión de salida. En particular, G_{vd0} es igual a:

$$G_{vd0} = \frac{< v_g >_{Tu}}{1- < d_a >_{Tu}{}^2} \tag{3.20}$$

La ganancia de la planta depende del punto de trabajo en el que se encuentre el convertidor, por lo que debe calcularse para las condiciones nominales. Sustituyendo (3.19) en (3.18), se obtiene la condición definitiva para evitar ciclo límite en régimen permanente:

$$G_{vd0} \cdot (1- < d_a >_{Tu}) \cdot H_v \cdot q_k < q_{ADC} \tag{3.21}$$

Sustituyendo la ecuación anterior con los datos del convertidor diseñado en este capítulo, el ciclo límite no aparece si se cumple la siguiente condición:

$$\begin{aligned} G_{vd0} \cdot (1 - < d_a >_{Tu}) \cdot H_v \cdot q_k &< q_{ADC} \\ 695{,}6522 \cdot (1 - 0{,}425) \cdot \frac{5}{500} \cdot q_k &< 5 \cdot \frac{1}{2^{10}} \\ q_k &< 0{,}0012 \end{aligned} \tag{3.22}$$

La condición de ausencia de ciclo límite se cumple si la resolución del regulador es suficientemente fina. El regulador diseñado utiliza 14 bits fraccionales, por lo que $q_k = 2^{-14} = 0{,}00006$ es mucho menor que 0,0012. Por tanto, la condición se cumple, y no se producirá ciclo límite en régimen permanente, ya que el regulador encontrará una zona de error nulo.

La condición para evitar ciclo límite en convertidores dc-dc [30, 29] también debe comprobarse en las técnicas PFC, ya que el comando de potencia k cambia el ciclo de trabajo medio durante un semiciclo de red. Sin embargo, el ciclo de trabajo medio se obtiene con todos los ciclos de trabajo dentro de un semiciclo de red, por lo que hay intrínsecamente una técnica de *dither* [30] al obtener el valor medio de ciclo de trabajo. Teniendo en cuenta esto, otra condición para evitar el ciclo límite régimen permanente se muestra en (3.23), la cual que será fácilmente satisfecha gracias al *dither* intrínseco al calcular el ciclo de trabajo medio durante un semiciclo de red.

$$\begin{aligned} G_{vd0} \cdot H_v \cdot \frac{q_{DPWM}}{\#sw_{cyc}} &< q_{ADC} \\ 695{,}6522 \cdot \frac{5}{500} \cdot \frac{q_{DPWM}}{1000} &< \frac{5}{2^{10}} \\ q_k &< 0{,}7 \end{aligned} \tag{3.23}$$

En la ecuación previa, q_{DPWM} es la resolución del PWM digital, y $\#sw_{cyc}$ es el número de ciclos de conmutación que hay durante un semiciclo de red, siendo igual a 1000 en el regulador propuesto. El regulador utiliza 15 bits para el PWM (ver sección 3.6), incluyendo 5 bits para realizar *dither* [30]. Por tanto, $q_{DPWM} = 2^{-15} = 0{,}00003 << 0{,}7$, por lo que la condición para evitar el ciclo límite debido a la resolución del ciclo de trabajo también es satisfecha.

Las condiciones vistas en 3.22 y 3.23 comprueban si hay una salida del regulador la cual resida en una zona de error nulo para el ADC. Sin embargo, el ciclo límite

también puede aparecer después de un transitorio si la parte integral del regulador es suficientemente alta como para que en un único ciclo de regulación se pueda cruzar esa zona de error nulo. La condición dinámica para evitar ciclo límite es:

$$G_{vk0} \cdot H_v \cdot K_i < 1 \tag{3.24}$$

donde K_i es la parte integral del regulador PID. En el sistema propuesto, la componente integral del regulador es 2^{-11}, y suficientemente baja como para cumplir la condición:

$$\begin{aligned} G_{vd0} \cdot (1 - < d_a >_{Tu}) \cdot H_v \cdot K_i &< 1 \\ 695{,}6522 \cdot (1 - 0{,}425) \cdot \frac{5}{500} \cdot 2^{-11} &< 1 \\ 0{,}002 &<< 1 \end{aligned} \tag{3.25}$$

Por tanto, el sistema tampoco llegará a una situación de ciclo límite tras un transitorio.

Como conclusión, a la hora de comprobar si puede existir ciclo límite en convertidores PFC, se debe cumplir la misma condición necesaria para convertidores dc-dc, es decir, que la resolución en el PWM digital sea más fina que la resolución del DC, pero usando la resolución efectiva del PWM durante un semiciclo de red, y por tanto teniendo en cuenta el *dither* intrínseco de los sistemas PFC. Además, para evitar ciclo límite, es necesario que la tensión de salida se muestre de forma sincronizada o al menos debe elegirse correctamente la ventana de medición del ADC de la tensión de salida. También para evitar ciclo límite, el comando de potencia k debe tener suficiente resolución, y la ganancia integral del regulador que genera k no puede ser demasiado alta. La resolución del PWM digital se ha estudiado profundamente en la literatura, mientras que la resolución del comando k es fácil de obtener, ya que es una variable interna del regulador y se puede generar con resolución arbitraria. La resolución del comando de potencia puede ser tan grande como se desee, pero no tendría sentido que fuera mucho más fina que la resolución del PWM, ya que el comando de potencia se usa finalmente para cambiar el ciclo de trabajo medio.

Tabla 3.1: Parámetros del convertidor boost construido para ciclo de trabajo precalculado

Parámetro	Valor
f_{sw}	100 kHz
$resolución_{PWM}$	1000 valores
L	5 mH
C	68 μF
P_{out}	300 W
V_{out}	400 V

3.6. Resultados

En esta sección se van a mostrar los experimentos realizados y los resultados obtenidos para comparar todos los métodos propuestos. De esta forma, se analizarán factores de potencia y armónicos obtenidos, cumplimiento de normativa, y coste *hardware* en términos de área y frecuencia máxima de funcionamiento.

Las pruebas de este capítulo se han basado en un convertidor elevador, construido con las características descritas en la tabla 3.1. Aunque el convertidor utilizado en este capítulo es diferente al utilizado durante las pruebas experimentales en el capítulo anterior, la principal diferencia radica en el aislamiento de las señales digitales de control respecto a la etapa de potencia, siendo el convertidor más robusto ante ruido electromagnético. Este ruido en su mayor medida está provocado por la conmutación del convertidor. En cualquier caso, el aislamiento no es una característica necesaria para hacer ciclo precalculado. La siguiente diferencia principal es el uso de un condensador de salida con capacidad ligeramente menor, 68 μF en vez de 100 μF. La menor capacidad en el condensador de salida implica mayor rizado en la tensión de salida, por lo que el lazo de rizado de la tensión de salida puede actuar con mayor resolución. Sin embargo, el cambio del condensador tampoco es significativo, habiendo probado que el funcionamiento con mayor capacidad es totalmente factible.

Todas las pruebas han sido realizadas con la FPGA Xilinx XC3S1000-4FT256. El reloj usado tiene una frecuencia de 50 MHz y su frecuencia ha sido doblada con un DCM (del inglés *Digital Clock Manager*), por lo que la frecuencia de trabajo es igual a 100 MHz. La frecuencia de conmutación es igual a 100 kHz, generando una señal PWM para el interruptor con valores entre 0 y 999, aunque internamente se han añadido 5 bits de dither [30]. Por otra parte, la frecuencia de red utilizada es de 50 Hz y, por tanto, igual a 100 Hz después de la rectificación. Teniendo en cuenta la frecuencia de la tensión de entrada y rectificada, y la frecuencia de conmutación, el semiciclo de red se divide en 1 000 ciclos de conmutación.

El cálculo de los ciclos de trabajo se ha automatizado realizando *scripts* en MathWorks Matlab, y realizando los cálculos con alta precisión. Estos valores calculados se guardan en la FPGA para que el sistema pueda aplicarlos en cada semiciclo de red. Dependiendo del método usado, los valores calculados son directamente el ciclo de trabajo en su valor complementario (en el primer método) o sus componentes por separado (en el caso de los dos casos restantes). En el primer método, el ciclo de trabajo completo se guarda en su formato complementario $(1 - d)$. Por su parte, el segundo método necesita leer los parámetros $(1 - d_1)$ y d_2. Finalmente, en el tercer método se almacenan los parámetros $(1 - d_a)$, $(1 - d_1)$ y d_c.

Los parámetros descritos se almacenan en las *block* RAMS de la FPGA. Cada valor para un ciclo de conmutación se guarda en 16 bits, usando 11 bits para almacenar un valor entre 0 y 999 con signo, y 5 bits para almacenar valores fraccionarios del ciclo de trabajo. Estos valores se usan para implementar la técnica de *dither*, la cual incrementa internamente la resolución del PWM. Teniendo en cuenta el tamaño de cada valor, y el número de valores, se extrae que cada componente almacenada necesita 16 000 bits. Este tamaño es menor que el ofrecido por una *block* RAM de la FPGA, ya que esta FPGA puede ser configurada con 24 módulos de 16 *kb* cada módulo.

Los reguladores para los dos lazos propuestos, lazo de tensión media de salida y lazo del rizado de la tensión de salida, son reguladores sencillos PID. Su implementación ha sido realizada usando coma fija, similar a la descrita en la sección 2.3.5. Como se ha comentado previamente, el lazo de tensión media de salida es similar a un lazo de tensión en un convertidor de factor de potencia clásico, por lo que tiene bajo ancho de banda. Por otra parte, el lazo del rizado de la tensión de salida se comporta como un lazo de corriente en un corrector clásico. Sin embargo, este lazo no mide la corriente de entrada, y tiene un ancho de banda bajo, ya que la entrada del regulador es el rizado de la tensión de salida durante un semiciclo de red.

El único ADC usado mide la tensión de salida en múltiples puntos dentro de un semiciclo de red. La FPGA calcula la media de las medidas como entrada del lazo de tensión media. Con el mismo ADC se calcula también el rizado como diferencia entre máximo y mínimo durante cada semiciclo de red. Dicho valor es la entrada al lazo de rizado de tensión que regula la carga de forma indirecta.

Por último, la aplicación del ciclo precalculado requiere que los datos guardados en las memorias y la corriente de entrada estén sincronizados. Para ello se usa un comparador de tensión, el cual compara la tensión de entrada, a través de un divisor resistivo, con una tensión de referencia. Cuando la tensión de entrada rectificada está por debajo de 10 V, el comparador fija su salida a '1' y, en caso contrario fija un '0'.

Tabla 3.2: Equipamiento usado para los resultados experimentales.

Equipamiento	Fuente de corriente alterna	Osciloscopio	Medición PFC (THDi y PF)
Equipamiento 1	Adaptive Power Systems FC200	Agilent MSO-X 3104A	FFT del osciloscopio
Equipamiento 2	Pacific 345-AMX	Tektronix MSO 2014	Analizador de potencia Voltech PM1000+

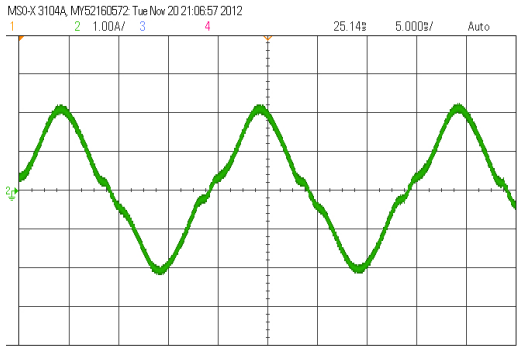
En la FPGA se ha implementado un sencillo filtro para eliminar los rebotes en la salida del comparador (ver sección 3.4). Este filtro genera la señal de sincronización entre el semiciclo de red y el direccionamiento de las memorias.

Los resultados han sido tomados con dos equipamientos de medición diferentes. Los dos equipamientos están resumidos en la tabla 3.2. El primer equipamiento incluye una fuente de corriente alterna de menores prestaciones, y las mediciones del factor de potencia y la distorsión armónica de la corriente de entrada han sido realizadas con la FFT (del inglés *Fast Fourier Transform*) analizando la forma de onda de la corriente de entrada. El segundo equipamiento consta de una fuente de corriente alterna más avanzada con posibilidad de generar tensiones distorsionadas, y el sistema de medición se realiza con un analizador de potencia, el cual tiene mayor precisión en sus medidas. En cada experimento se detallará qué equipamiento se ha usado para obtener los resultados experimentales.

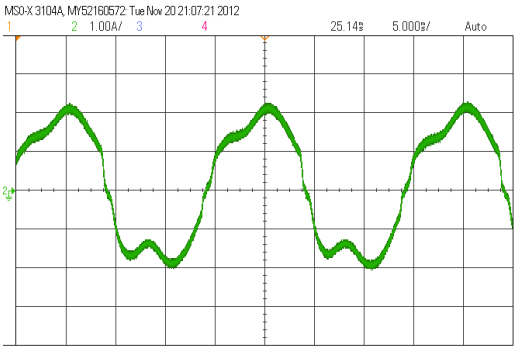
3.6.1. Comparativa entre regular el ciclo de trabajo d o su complementario $(1-d)$

En la sección 3.3.1, el ciclo de trabajo se trata como una componente única, la cual es modificada según el valor medio en la tensión de salida. Se describió e ilustró con un ejemplo el deterioro en la forma del ciclo de trabajo cuando se regula directamente. Sin embargo, regulando su término complementario, $1-d$, el ciclo de trabajo apenas se desvía del ciclo de trabajo regulado idealmente.

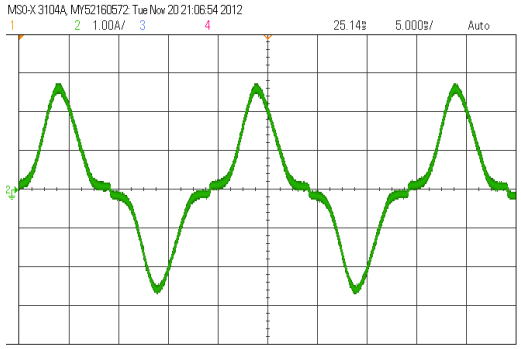
A continuación se muestran pruebas experimentales sobre estas tres posibilidades de regulación usando el equipamiento 1 (ver tabla 3.2). Se ha probado el método de la sección 3.3.1, utilizando las tres regulaciones. En todos los sistemas, se ha variado la tensión de entrada para que el regulador tenga que actuar un 3 % por encima y debajo de su valor nominal. Se han extraído las formas de onda de la corriente de entrada, así como datos numéricos de la distorsión armónica de la corriente (THD) y del factor de potencia. En la figura 3.24 se muestran la corriente de entrada durante un semiciclo de red. La figura muestra que el método de regulación sobre $(1-d)$ es mucho más robusto ante actuaciones tanto positivas como negativas. Por otra parte, la tabla 3.3



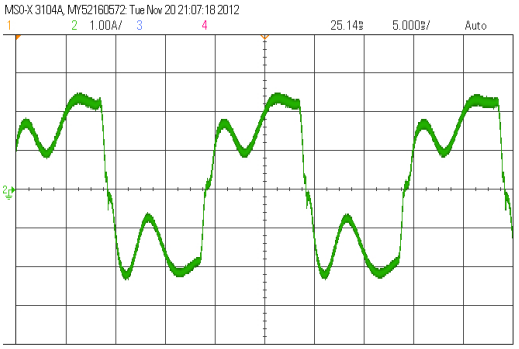
(a) Método $d + \delta_1$. Regulación del $-3\,\%$.



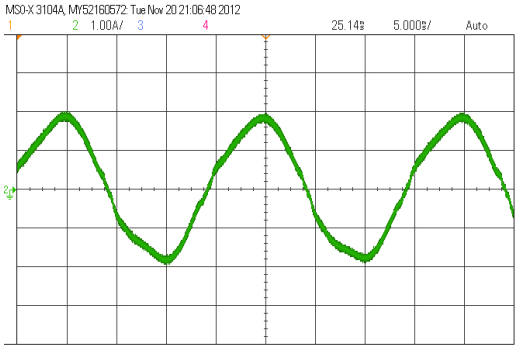
(b) Método $d + \delta_1$. Regulación del $+3\,\%$.



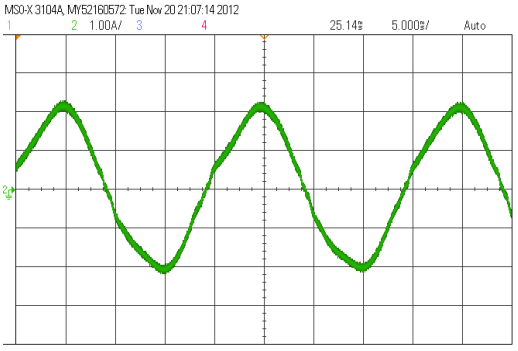
(c) Método $d \cdot k_2$. Regulación del $-3\,\%$



(d) Método $d \cdot k_2$. Regulación del $+3\,\%$



(e) Método $1 - (1 - d) \cdot k_3$. Regulación del $-3\,\%$.



(f) Método $1 - (1 - d) \cdot k_3$. Regulación del $+3\,\%$.

Figura 3.24: Corriente de entrada regulando $d + \delta_1$, $d \cdot k_2$ y $1 - (1 - d) \cdot k_3$.

muestra la distorsión armónica de la corriente y el factor de potencia obtenido con ambos métodos Los resultados numéricos también muestran claramente la ventaja de regular $(1 - d)$, obteniendo factores de potencia por encima de 0,99 en ambos casos. Queda demostrado que **la regulación con $(1 - d)$ es más precisa**, por lo que el método descrito en la sección 3.3.1 usa el término complementario del ciclo de trabajo.

Tabla 3.3: Factor de potencia y distorsión armónica regulando d y $(1-d)$.

Método	Regulación +3 %		Regulación −3 %	
	PF	**THDi**	**PF**	**THDi**
Método $D+\delta$:	0,97287	16,6994 %	0,95655	21,31376 %
Método $d \cdot k_2$	0,87327	38,0949 %	0,77220	54,3135 %
Método $1-(1-d)\cdot k_3$	0,99465	7,3373 %	0,99467	7,3233 %

Tabla 3.4: Resultados de implementación de los tres métodos para la FPGA Xilinx XC3S1000.

Method	LUTs de 4 entradas	*Flip flops*	Multiplicadores 18x18	*Block RAM* (16 *kb*)
Método 1: d	165 (1,07 %)	91 (0,59 %)	2 (8,33 %)	1 (4,17 %)
Método 2: d_1, d_2	245 (1,60 %)	109 (0,71 %)	5 (20,83 %)	2 (8,33 %)
Método 3: d_a, d_b, d_c	309 (2,01 %)	109 (0,71 %)	6 (25 %)	3 (12,5 %)
Sincronización, PWM, Interfaz ADC, etc	828 (5,39 %)	457 (2,98 %)	2 (8,33 %)	0 (0 %)

3.6.2. Comparativa de los métodos de regulación propuestos

En este apartado se muestran los resultados experimentales, usando el equipamiento 1 (tabla 3.2) relacionados con los tres métodos descritos en las secciones 3.3.1, 3.3.2 y 3.3.3.

En primer lugar se han implementado los tres sistemas en la FPGA comentada anteriormente. La tabla 3.4 muestra los resultados de implementación más significativos, dando los resultados de cada regulador aislado y del resto de lógica necesaria para controlar el ADC, la sincronización con la red eléctrica, generación del PWM, etc. Como se puede observar, la mayoría de las LUTs y *flip flops* se usan en la etapa de sincronización con la red, el cálculo del PWM con *dither*, en la interfaz con el ADC, etc. Ninguno de los tres métodos propuestos usan una cantidad de recursos excesiva siendo, en todo caso, inferior a los recursos del resto del sistema. Por tanto, la decisión sobre qué método usar no debe tomarse basándose en los recursos necesarios.

Se han realizado experimentos para comprobar el comportamiento de los diferentes métodos ante cambios en las condiciones de entrada. El primer experimento ha consistido en la variación de la tensión de entrada. En particular, se ha subido y bajado un 10 % la tensión de entrada, para comprobar el comportamiento de la corriente de entrada. Los resultados se muestran en la figura 3.26. La fila superior corresponde a tensiones de entrada 10 % menores a la nominal, la fila central muestra los resultados en condiciones nominales, y la fila inferior muestra los resultados correspondientes a tensiones de entrada 10 % mayores a la nominal. Como se puede observar, la corriente de entrada es similar usando los tres métodos. Todos los métodos **se comportan de**

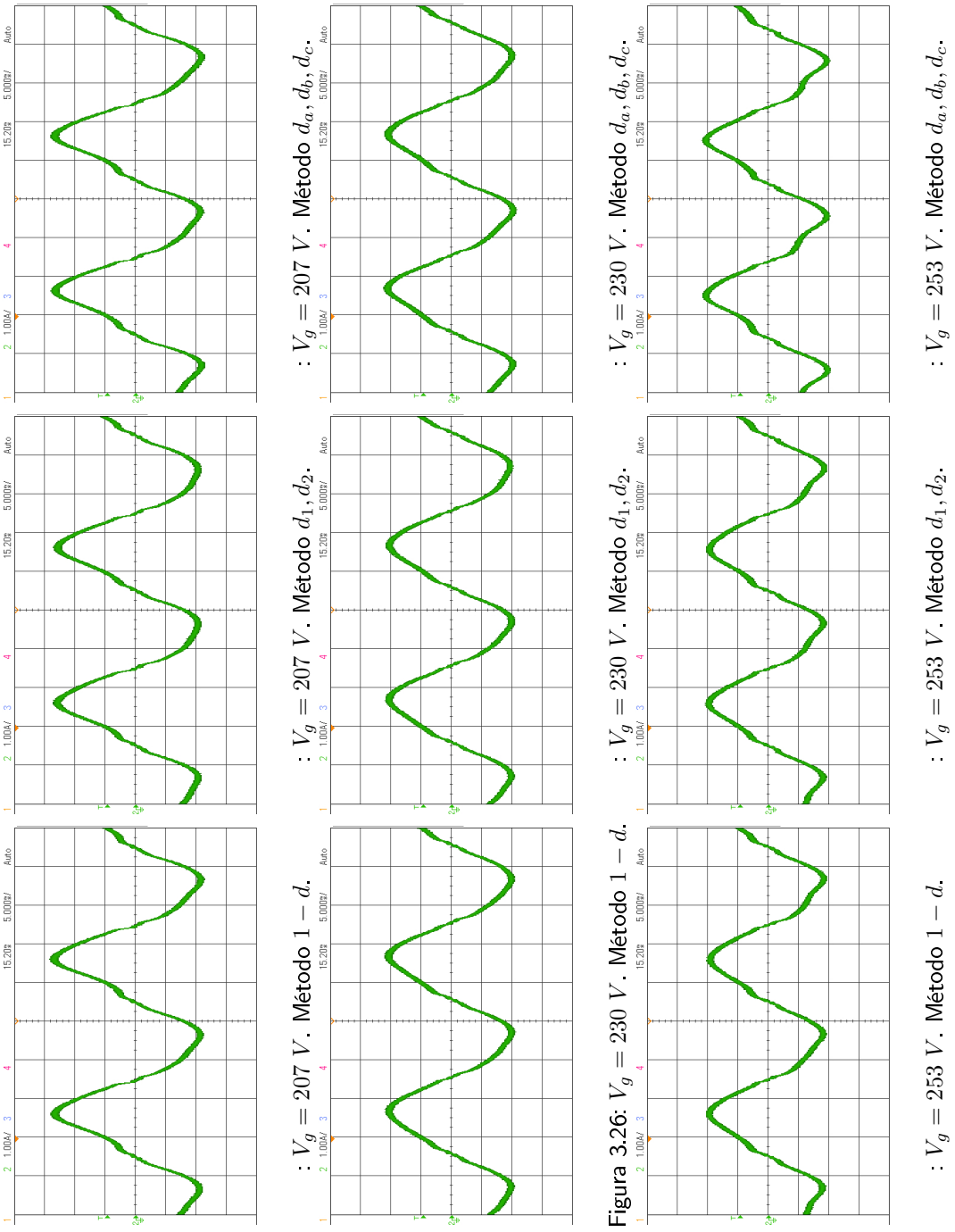


Figura 3.26: $V_g = 230\ V$. Método $1 - d$.

Figura 3.26: Corriente de entrada frente a diferentes tensiones de entrada.

Tabla 3.5: Factor de potencia y distorsión armónica ante cambios en la tensión de entrada. V_g nominal igual a 230 V.

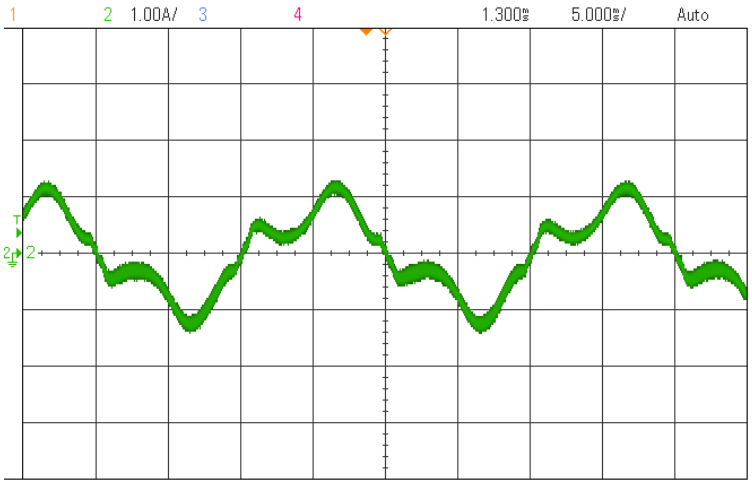
Método	207 V		230 V		253 V	
	PF	**THDi**	**PF**	**THDi**	**PF**	**THDi**
d	0,97906	14,6257 %	0,99450	7,4394 %	0,98910	10,4974 %
d_1, d_2	0,98336	13,0075 %	0,99442	7,4893 %	0,98596	11,9337 %
d_a, d_b, d_c	0,97968	14,4034 %	0,993	9,30 %	0,98075	14,0104 %

forma similar ante cambios de tensión de entrada. Este comportamiento es lógico ya que el ciclo de trabajo está altamente influenciado por la relación entre la tensión de entrada y la tensión de salida, y todos los métodos tienen en cuenta dicha relación. La tabla 3.5 muestra los resultados numéricos de la prueba, tanto en factor de potencia, como en distorsión armónica de la corriente.

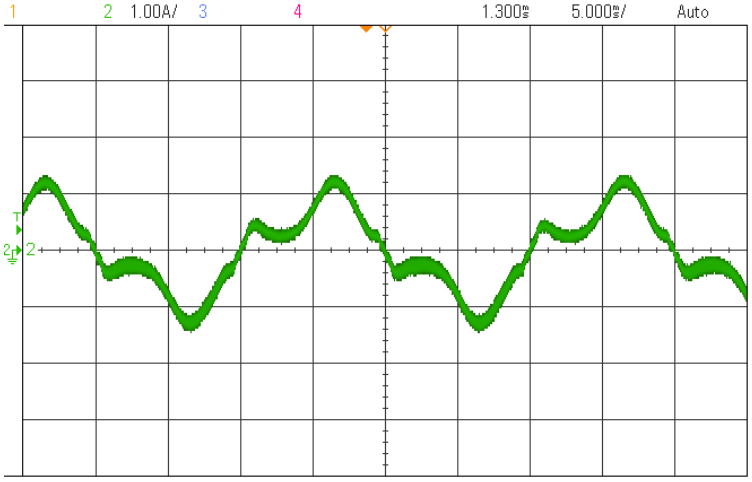
El experimento anterior consistía en cambiar la tensión de entrada. Los tres métodos se comportan de forma muy parecida ante dicho cambio, ya que es fácil corregir el error producido por tensiones de entrada no nominales. Además es importante destacar que los cambios en la tensión de entrada son muy pequeños en situaciones reales.

Por otra parte, los cambios en la carga del convertidor sí son muy comunes y mucho mayores. Por ello se han realizado experimentos para probar los tres sistemas con diferentes cargas. De esta forma, el lazo de rizado de la tensión de salida puede ser analizado. El lazo de tensión media también está funcionando, ya que siempre es necesario controlar la tensión media de salida, pero su actuación es muy leve ya que la tensión media apenas cambia con diferentes cargas. En los tres métodos, el precálculo del ciclo de trabajo se ha realizado para las condiciones nominales, con $P = 300\ W$. La figura 3.28 muestra el factor de potencia para los tres métodos con diferentes cargas, todos en régimen permanente. Además, la figura 3.27 muestra la forma de la corriente en el mismo experimento para un caso concreto, con el 50 % de carga respecto a la nominal. Se observa claramente que **el tercer método (d_a, d_b, d_c) obtiene los mejores resultados cuando la carga no es la nominal.**

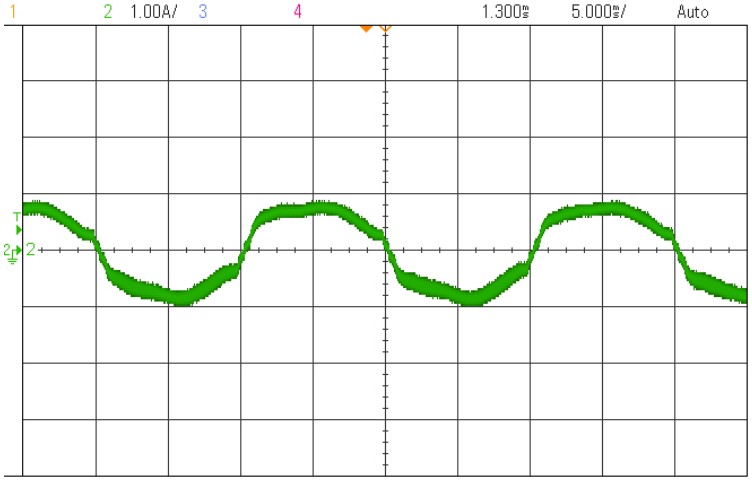
Podría pensarse que el segundo método (d_1, d_2) es mejor que el primer método (d), porque el segundo método regula d_2 con el rizado de la tensión de salida, mientras que el primer método no lo hace. Sin embargo, el segundo método no tiene en cuenta la componente de rizado dentro del parámetro d_1, y la regulación de d_2 sin tener en cuenta el rizado en d_1 es contraproducente. Esto puede ser explicado mirando la figura 3.17(b), la cual muestra d_b, que realmente es la distorsión por el rizado de la componente d_1. Comparando las componentes d_2 y d_b en las figuras 3.13(b) y 3.17(b) respectivamente, se observa que d_b es mayor que d_2, y regulando sólo d_2, **el**



(a) Método d.



(b) Método d_1, d_2.



(c) Método d_a, d_b, d_c.

Figura 3.27: Corriente de entrada con potencia $P = 147\ W$ (Precalculado para $300\ W$).

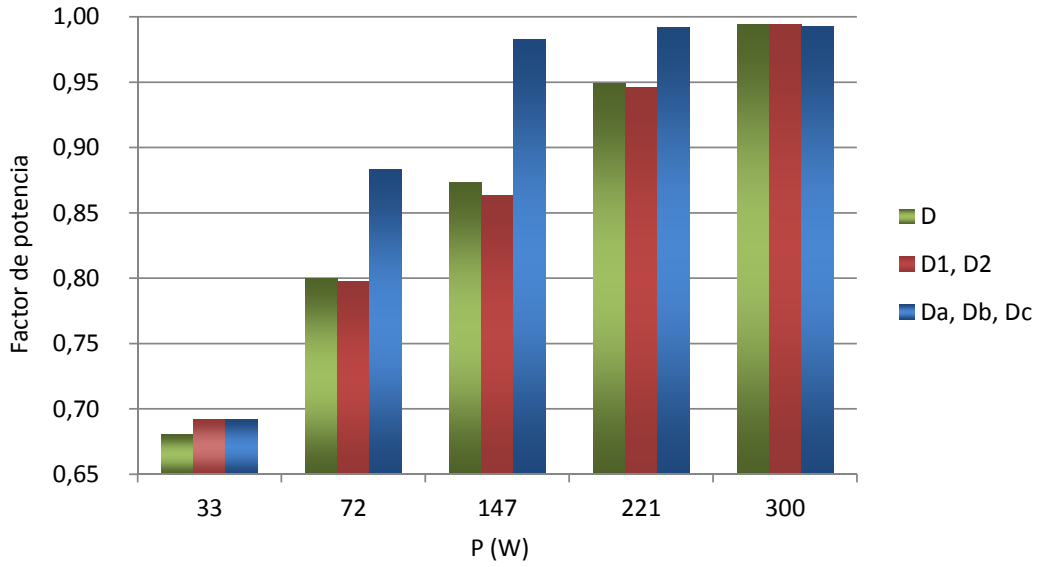


Figura 3.28: Factor de potencia de todos los métodos para diferentes cargas. Ciclos de trabajo precalculados para $Vg = 230\ V$, $P = 300\ W$ y $V_{out} = 400\ V$.

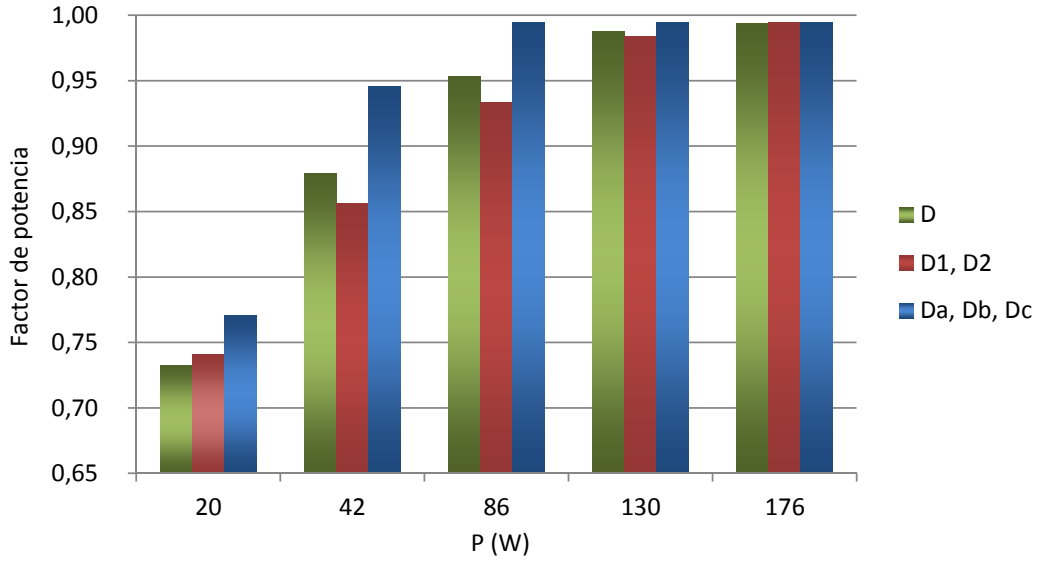


Figura 3.29: Factor de potencia para los tres métodos con diferentes cargas. Ciclos de trabajo precalculados para $Vg = 120\ V$, $P = 176\ W$ y $V_{out} = 300\ V$.

segundo método obtiene peores resultados que regulando únicamente la componente d.

El mismo experimento también se ha realizado para otro conjunto de datos precalculados: $V_g = 120\ V$, $V_{out} = 300\ V$ y $P = 176\ W$. De esta forma, podemos ver si los métodos se comportan de forma similar con diferentes tensiones y potencias.

Tabla 3.6: Normativa IEC 61000-3-2.

Armónico	**Clase A (A)**	**Clase B (A)**	**Clase C (% del fundamental)**	**Clase D (mA/W)**
Armónicos impares				
3^o	2,30	3,45	30· factor de potencia	3,4
5^o	1,14	1,71	10	1,9
7^o	0,77	1,155	7	1,0
9^o	0,40	0,60	5	0,5
11^o	0,33	0,495	3	0,35
13^o	0,21	0,315	3	3,85/13
$15^o \leq n \leq 39^o$	$0{,}15 \cdot 15/n$	$0{,}225 \cdot 15/n$	3	$3{,}85/n$
Armónicos pares				
2^o	1,08	1,62	2	-
4^o	0,43	0,645	-	-
6^o	0,3	0,45	-	-
$8^o \leq n \leq 40^o$	$0{,}23 \cdot 8/n$	$0{,}345 \cdot 8/n$	-	-

La figura 3.29 muestra los resultados para estas condiciones. En general todos los métodos obtienen mejores resultados con menores tensiones. Con estas condiciones, el tercer método (d_a, d_b, d_c) sigue obteniendo resultados mucho mejores, mientras que el segundo método (d_1, d_2) obtiene los peores.

Por último, se han hecho pruebas de cumplimiento de normativa con los tres métodos propuestos. En particular se ha tenido en cuenta la normativa de contenido armónico IEC 61000-3-2, publicada por la *International Electrotechnical Comission* (IEC). Esta normativa limita el contenido armónico en la red eléctrica causado por el equipamiento conectado a la red. Este estándar define cuatro clases de equipamiento, y limita de forma diferente cada clase. La clase B agrupa el equipamiento portátil y los sistemas de soldadura por arco. Por su parte, la clase C define las limitaciones de los equipos de iluminación y es la más restrictiva. La clase D agrupa los equipamientos por debajo de 600 W que pueden provocar gran impacto en la red eléctrica. En otros, ésta última clase engloba a los ordenadores personales, televisiones y monitores. Por último, la clase A engloba a todo el equipamiento no recogido por las clases anteriores. La tabla 3.6 resume las limitaciones de cada clase.

Todas las clases han sido probadas (A, B, C y D), y todos los métodos han superado las pruebas. La tabla 3.7 muestra los resultados de cumplimiento de normativa para el tercer método (d_a, d_b, d_c). Como se puede observar, todos los armónicos analizados **son mucho menores a los límites que determina la normativa citada**. En particular, en esta prueba se ha usado el equipamiento 2, el cual incluye un analizador de potencia Voltech PM1000+. Por motivos de espacio, no se han mostrado todos los armónicos involucrados en la citada normativa. Sin embargo, el analizador de potencia usado permite realizar pruebas completas de cumplimiento de normativa, y el sistema cumple la normativa para todas las clases.

Tabla 3.7: Prueba del método D_a, D_b, D_c para la clase C de la normativa IEC 61000-3-2.

Armónico	**Valor medido (A)**	**Máximo Clase A (A)**	**Máximo Clase B (A)**	**Máximo Clase C (A)**	**Máximo Clase D (A)**
1^o	1,39				
2^o	0,0035	1,080	1,620	0,028	-
3^o	0,135	2,300	3,450	0,414	1,020
5^o	0,045	1,140	1,710	0,139	0,570
7^o	0,020	0,770	1,155	0,097	0,300

3.6.3. Pruebas de dinámica

El presente capítulo ha mostrado diferentes formas de regulación del ciclo de trabajo precalculado para corrección de factor de potencia. Las citadas técnicas en mayor o menor medida permiten obtener los ciclos de trabajo óptimos para condiciones de trabajo diferentes a las nominales, y en régimen permanente. En el capítulo se ha comentado que dichas técnicas confían su dinámica en reguladores de tipo PID. El objetivo del capítulo no es mejorar la dinámica de reguladores para corrección de factor de potencia y, por ello, se han implementado reguladores muy estables y por tanto no muy rápidos. En cualquier caso, en esta sección se va a analizar la dinámica de los reguladores implementados.

Los últimos dos métodos propuestos (d_1, d_2 y d_a, d_b, d_c) utilizan dos lazos de control, uno midiendo la tensión medida de salida, y el otro midiendo el rizado de la tensión de salida.

El primer lazo controla la tensión media de salida, a través de d_a, la cual varía principalmente ante cambios de la tensión de entrada. La planta a regular, $G_{V_{out}K_1}$, expresa la relación entre la tensión de salida, V_{out}, y la salida del primer lazo, K_1:

$$\frac{V_{out}(s)}{K_1(s)} = \frac{V_{out}(s)}{D_a(s)} \cdot \frac{D_a(s)}{K_1(s)} \tag{3.26}$$

El primer elemento relaciona la tensión de salida con el ciclo de trabajo aplicado. La función de transferencia del primer elemento tiene la dinámica de un corrector de factor de potencia de técnica clásico, la cual está descrita en [28]: $\frac{1}{\frac{RC}{2}S+1}$. Por otra parte, la ganancia se puede calcular teniendo en cuenta la siguiente relación:

$$v_{out} = \frac{v_{in}}{1 - d_a} \tag{3.27}$$

Donde todos los valores serán los eficaces a lo largo de un semiciclo de red. Para calcular la ganancia, se debe linealizar en torno al punto de equilibrio:

$$\hat{v}_{out} = \frac{V_{in}}{(1-D_a)^2}\hat{d}_a \tag{3.28}$$

Donde V_{in} y D_a son los valores eficaces a lo largo de un semiciclo de red. Por tanto, la relación $\frac{V_{out}(s)}{D_a(s)}$ teniendo en cuenta tanto su ganancia como su dinámica es:

$$\frac{V_{out}(s)}{D_a(s)} = \frac{V_{in}}{(1-D_a)^2}\frac{1}{\frac{RC}{2}s+1} \tag{3.29}$$

$$\tag{3.30}$$

Por otra parte, el término $\frac{D_a(s)}{K_1(s)}$ es la ganancia desde el comando K_1 hasta el ciclo de trabajo. La relación entre K_1 y $D_a(s)$ tiene dinámica, pero ésta se puede despreciar, ya que esa transformación se realiza cada ciclo de conmutación y, en cambio, se va a discretizar la planta con periodo igual al semiciclo de red. Por ello, se puede ignorar la dinámica y utilizar únicamente su ganancia. Como se ve en la figura 3.18, la componente d_a regulada se obtiene con la fórmula:

$$d_a* = 1 - k_1(1-d_a) \tag{3.31}$$

De nuevo se obtendrá la ganancia linealizando en torno al punto de equilibrio:

$$\hat{d}_a = -(1-D_a)\hat{k}_1 \tag{3.32}$$

La ganancia de $\frac{D_a(s)}{K_1(s)}$ es, por tanto:

$$\frac{D_a(s)}{K_1(s)} = -(1-D_a) \tag{3.33}$$

Como se puede observar en la ecuación anterior, la ganancia es negativa, por lo que el regulador también tendrá signo negativo. Una vez calculadas las dos componentes de la ecuación (3.26), se obtiene la planta completa a regular:

$$\frac{V_{out}(s)}{K_1(s)} = \frac{V_{in}}{(1-D_a)^2}\frac{1}{\frac{RC}{2}s+1}\cdot -(1-D_a)$$
$$\frac{V_{out}(s)}{K_1(s)} = -\frac{V_{in}}{(1-D_a)}\frac{1}{\frac{RC}{2}s+1} \tag{3.34}$$

Los valores de V_{in}, D_a, R y C pueden ser despejados, obteniendo la planta en el dominio continuo:

$$\frac{V_{out}(s)}{K_1(s)} = G_{V_{out}K_1}(s) = -\frac{400}{0{,}01813s+1} \tag{3.35}$$

Esta planta se ha discretizado para realizar un controlador diseñado directamente en el dominio discreto. Se ha utilizado un tiempo de muestro igual a un semiciclo de red (10 ms), y se ha discretizado usando el método de bloqueador de orden cero (en inglés Zero Order Hold):

$$G_{V_{out}K_1}(z) = -\frac{169{,}6}{z-0{,}5761} \tag{3.36}$$

Se ha diseñado un regulador PI para controlar la planta descrita:

$$R_{K_1V_{out}}(z) = -\frac{2^{-9}\cdot(z-0{,}75)}{z-1} \tag{3.37}$$

El regulador diseñado es un PI (proporcional-integral) y es muy conservador, premiando la estabilidad ante la rapidez de la actuación. El ancho de banda del regulador es de 3,7 Hz, no alejándose de los correctores de factor de potencia clásicos. En la figura 3.30 se muestra la respuesta en lazo cerrado del primer control ante un escalón en la tensión de salida.

El segundo lazo adapta el ciclo de trabajo que se aplica en función de la potencia demandada por la carga conectada al convertidor. La potencia demandada hace que el rizado de la tensión de salida cambie. Analizando la ecuación (3.6) durante un semiciclo de red, el rizado de la tensión de salida es igual a:

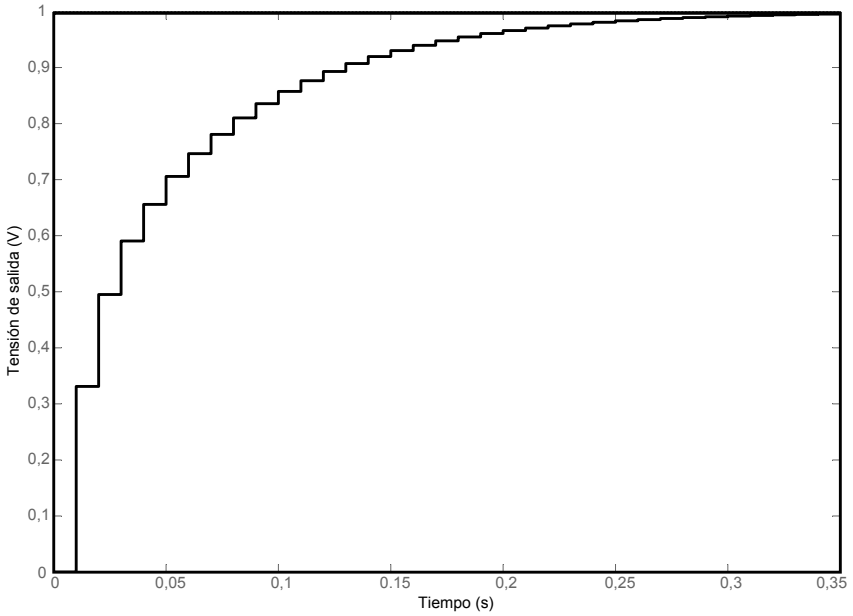


Figura 3.30: Respuesta en lazo cerrado del lazo de tensión media de salida ante un escalón de tensión.

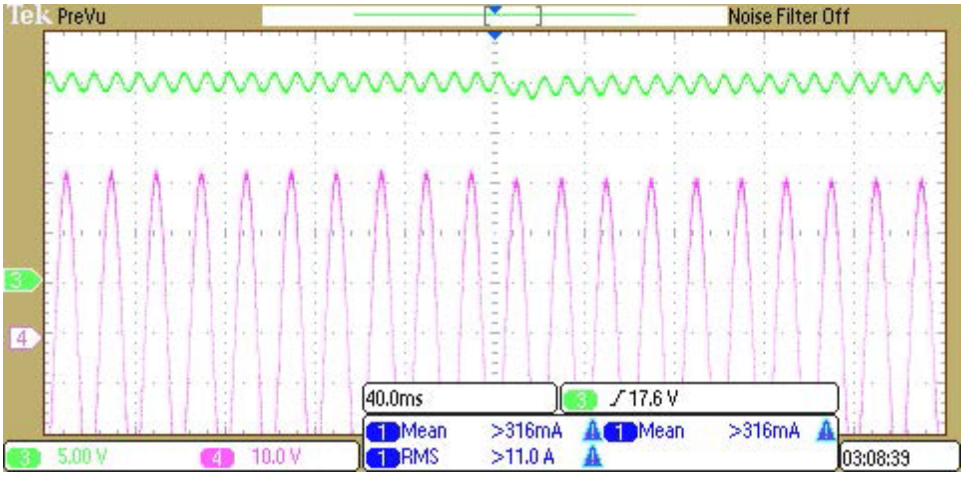
$$v_{riz} = \frac{P_{out}}{C \cdot \omega_r \cdot V_{out}} \tag{3.38}$$

El rizado de la tensión de salida no tiene dinámica, sino que obtiene su nuevo valor de forma instantánea, teniendo en cuenta que el rizado se define entre el valor máximo y mínimo dentro de un semiciclo de red. El segundo lazo lo único que hace es generar un valor K_2 proporcional al rizado detectado y, por tanto, proporcional a la carga demandada. El regulador, siendo únicamente proporcional es:

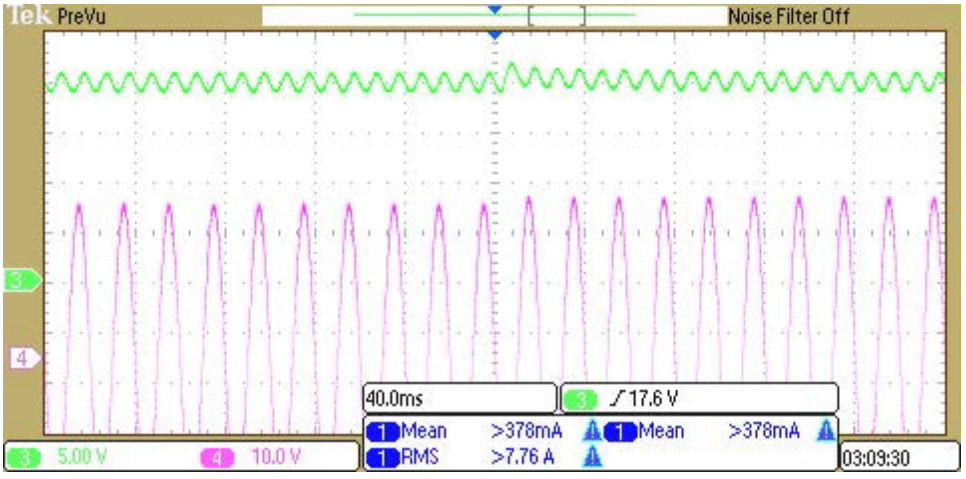
$$R_{K_2 V_{riz}} = \frac{1}{v_{rizado\ esperado}} \approx \frac{1}{36{,}73} \tag{3.39}$$

Se han realizado pruebas de dinámica aplicando transitorios, cambiando la tensión de entrada y la carga del convertidor. Todas las pruebas de dinámica se han realizado con el tercer método de regulación propuesto (d_a, d_b, d_c), pues que se ha visto que es el que ofrece mejores resultados, y usando el equipamiento 2 (ver tabla 3.2).

La figura 3.31(a) muestra un transitorio en el cual la tensión de entrada ha cambiado de 230 V a 220 V. Por otra parte, en la figura 3.31(b) se ve la respuesta del



(a) Cambio de la tensión de entrada de 230 V a 220 V.

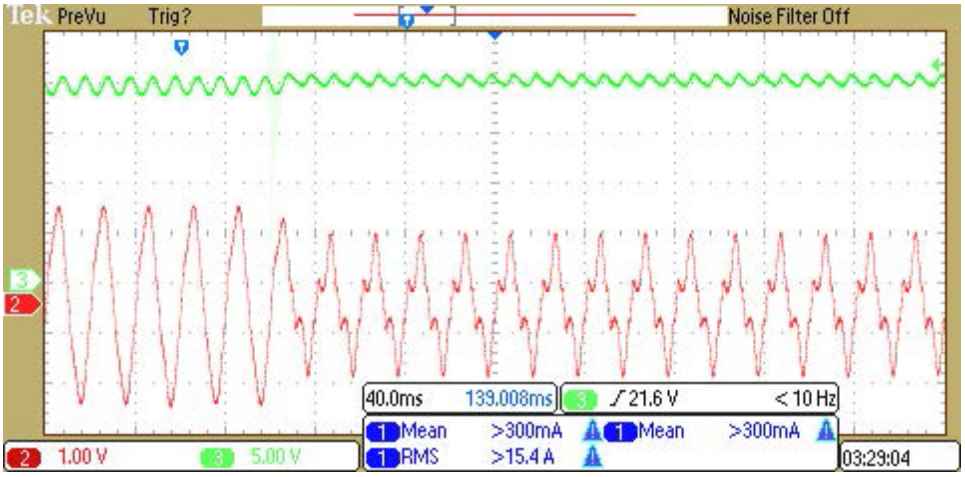


(b) Cambio de la tensión de entrada de 220 V a 230 V.

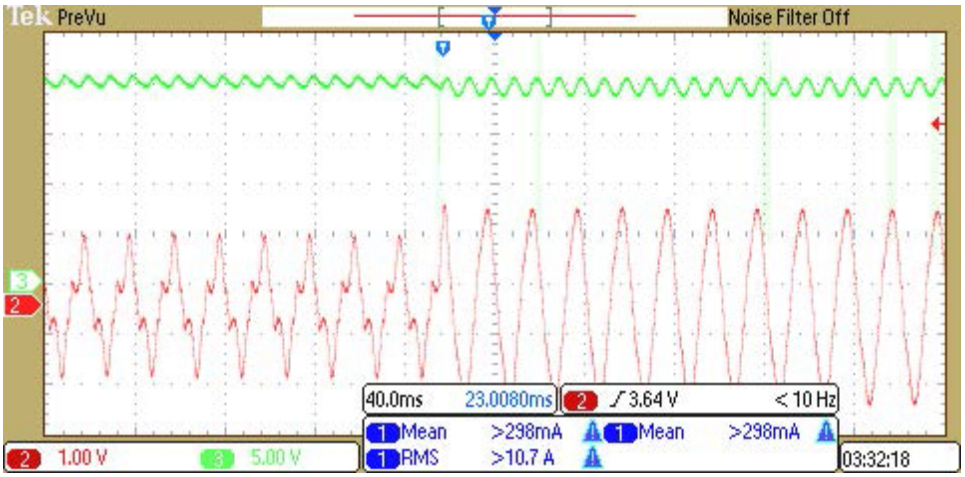
Figura 3.31: Transitorios al cambiar la tensión de entrada. Morado: Tensión de salida. Verde: Tensión de entrada.

sistema cuando la tensión de entrada ha cambiado de 220 V a 230 V. Como se puede observar, aproximadamente en nueve semiciclos de red, unos 90 ms, la tensión de salida vuelve a llegar al valor de referencia.

Por otra parte, se han hecho transitorios provocados por cambios en la carga. En la figura 3.32(a) se muestra el transitorio al reducirse la carga un 50 % (de 300 W a 150 W). En cambio, la figura 3.32(b) muestra un escalón de carga del 50 % al 100 %. Las figuras muestran que la acción del lazo es prácticamente instantánea. Es cierto que al 50 % de carga, la corriente de entrada queda notablemente distorsionada, pero la dinámica es rápida.



(a) Cambio de la tensión de entrada de 230 V a 220 V.



(b) Cambio de la tensión de entrada de 220 V a 230 V.

Figura 3.32: Transitorios al cambiar la carga del convertidor. Morado: Tensión de salida. Azul: Corriente de entrada

3.6.4. Influencia de la inductancia y conductancia del convertidor

Las técnicas de corrección de factor de potencia propuestas usan ciclos de trabajo precalculados. Como se ha visto durante este capítulo, estos cálculos se realizan para unas determinadas condiciones de trabajo, y cualquier cambio en el convertidor real afectará negativamente a la corrección de factor de potencia. Todos los sistemas propuestos se basan en la ecuación 3.5, la cual depende de la inductancia de la bobina, L. Por otra parte, el ciclo de trabajo depende indirectamente de la capacidad del condensador de salida del convertidor, C. Estas dos dependencias se ven resumidas en la siguiente ecuación:

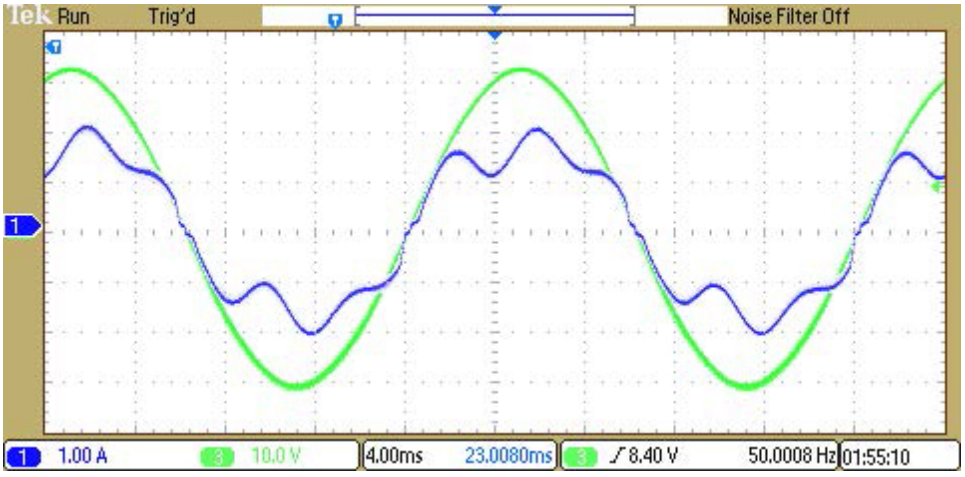
$$d(k) = \frac{v_{out}(k) - v_g(k)}{v_{out}(k)} + \frac{L}{T_{Sw}} \cdot \frac{(i_L(k+1) - i_L(k))}{v_{out}(k)}$$
$$v_{out}(k) = V_{out} - \frac{P_{out}}{C \cdot 2 \cdot \omega_r \cdot V_{out}} \cdot \sin(2 \cdot \omega_r \cdot k \cdot T_{sw}) \quad (3.40)$$

Siempre habrá una discrepancia entre los valores de C y L que se usaron en el precálculo, y los valores reales del convertidor. Por tanto, es necesario cuantificar el error que se produciría en la corrección de factor de potencia cuando estos errores ocurran. Para comprobar estas dependencias, se han realizado pruebas en las que los parámetros C y L han sido incrementados o decrementados un 10 % respecto a su valor real en el convertidor. Los resultados muestran que la variación de un 10 % hace que el THD de la corriente de entrada aumente en 0,96 %. Por otra parte, la misma variación en la capacidad del condensador de salida genera en el THD de la corriente de entrada un incremento de 8,51 %. Como se puede observar, la precisión en la estimación de la inductancia no es crítica para la corrección de factor de potencia. La estimación de la capacidad sí es más influyente en la corrección, generando un error del 0,8 % por cada 1 % de error en dicha estimación. En la ecuación 3.40 se puede ver que el valor de C influye no sólo en la segunda componente (llamada d_2 o d_c), sino también en la primera (d_1), teniendo esta última componente un peso mucho mayor en el ciclo de trabajo definitivo. Los resultados apoyan esta mayor importancia de la precisión del parámetro C en la corrección de factor de potencia. En cualquier caso, un incremento del THD menor al 1 % por cada 1 % de error en la estimación tampoco convierte en crítica dicha estimación.

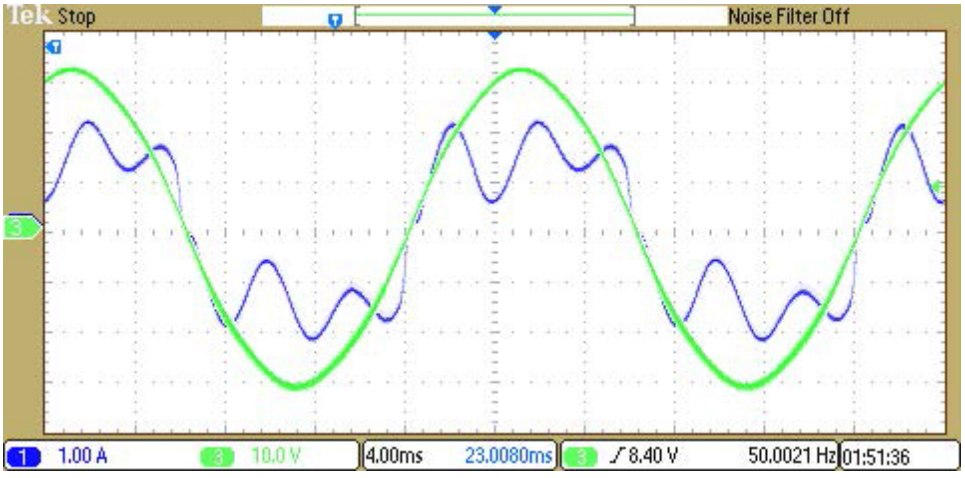
3.6.5. Resultados frente a tensión de entrada distorsionada

Todos los métodos propuestos utilizan un único ADC para realizar la corrección de factor de potencia. Debido a ello, la tensión de entrada no se mide, sino que sólo se detecta su paso por cero para que el sistema pueda sincronizarse con la red eléctrica. Dado que no se mide la tensión de entrada, el sistema propuesto es muy sensible a distorsiones en la tensión de entrada. Dependiendo de la instalación eléctrica, es habitual la presencia del tercer y quinto armónico en la tensión de entrada del convertidor. Por ello, se han realizado pruebas para ver el grado de sensibilidad del sistema precalculado frente a estas situaciones, usando el equipamiento 2 (ver tabla 3.2).

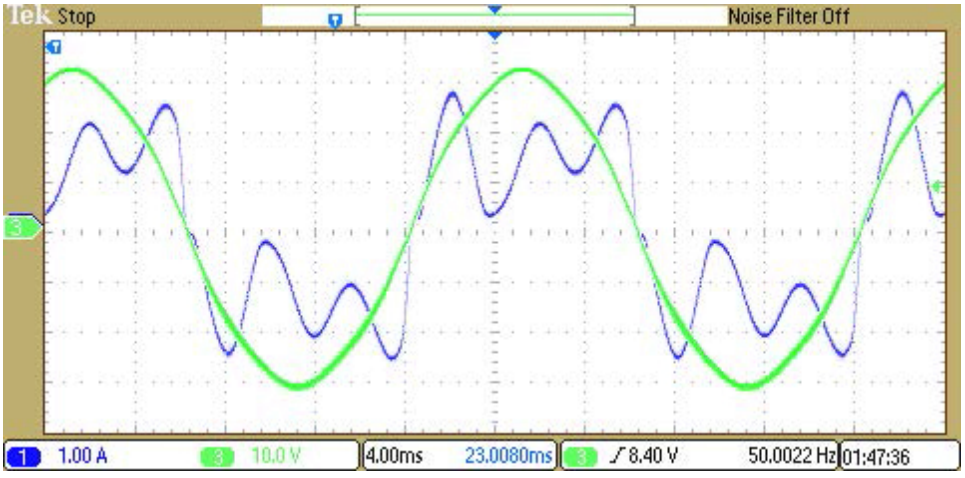
La figura 3.33 muestra la tensión de entrada y la corriente de entrada cuando la tensión de entrada contiene armónicos tercer y quinto con diferentes valores. La figura citada muestra el sistema con tensión de entrada igual a 230 V, tensión de salida igual



(a) Tercer y quinto armónico del $1\,\%$.

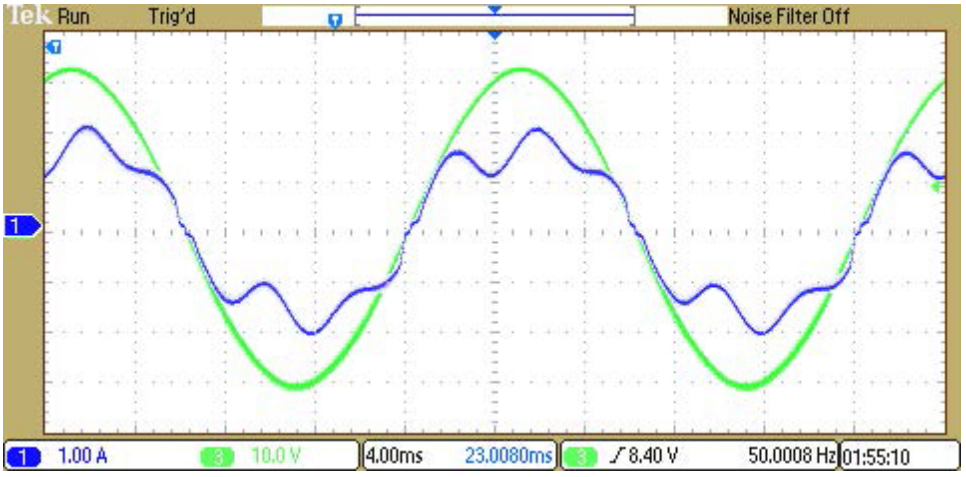


(b) Tercer y quinto armónico del $2\,\%$.

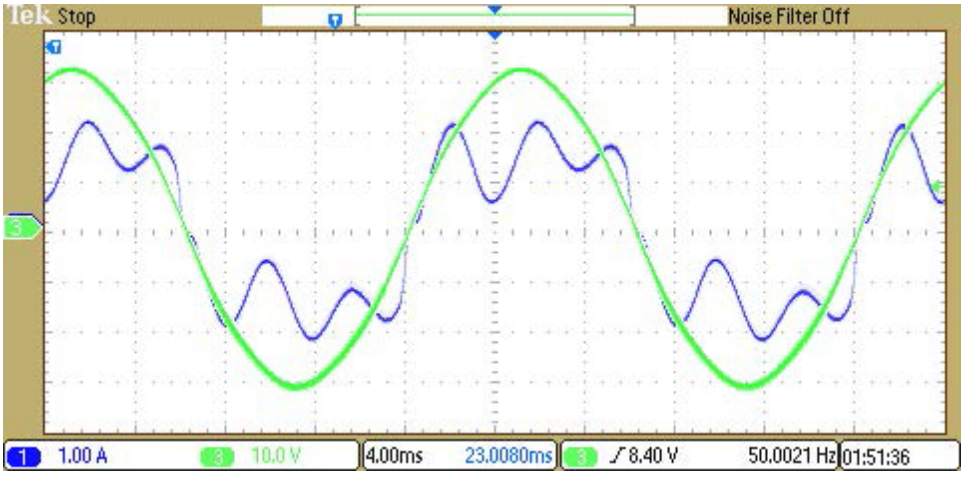


(c) Tercer y quinto armónico del $3\,\%$.

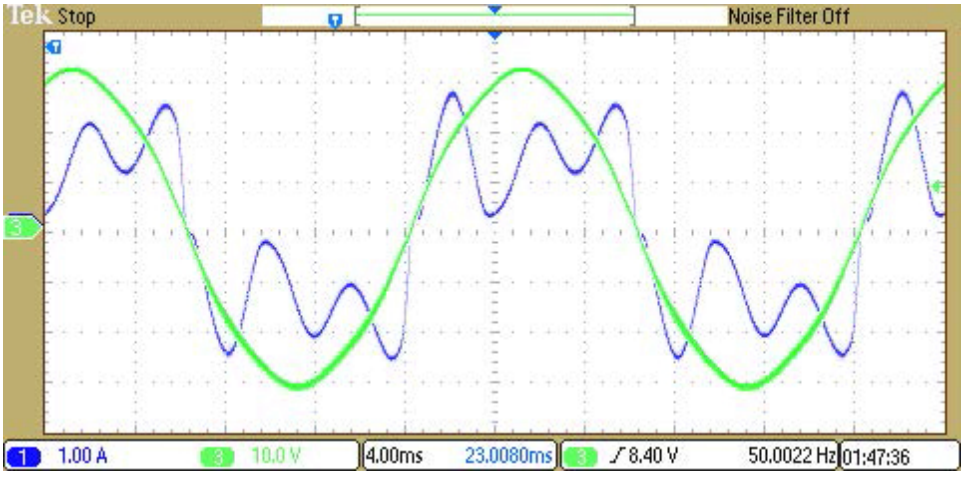
Figura 3.33: Corrección de factor potencia ante tensión de entrada distorsionada ($230\ V$). Morado: tensión de entrada. Amarilla: Corriente de entrada.



(a) Tercer y quinto armónico del $1\,\%$.



(b) Tercer y quinto armónico del $2\,\%$.



(c) Tercer y quinto armónico del $3\,\%$.

Figura 3.34: Corrección de factor potencia ante tensión de entrada distorsionada ($120\,V$). Morado: tensión de entrada. Amarilla: Corriente de entrada.

Tabla 3.8: Factor de potencia y distorsión armónica cuando la tensión de entrada contiene los armónicos tercero y quinto.

Armónicos 3^o y 5^o	$V_g = 120\ V$, $176\ W$		$V_g = 230\ V$, $300\ W$	
	PF	THDi	PF	THDi
0 %	0,997	7,20 %	0,993	9,30 %
1 %	0,995	8,00 %	0,972	24,00 %
2 %	0,979	20,40 %	0,901	48,00 %
3 %	0,951	33,70 %	0,810	73,00 %

a 400 V y potencia igual a 300 W. Los armónicos tercero y quinto se muestran en relación al armónico fundamental. Por otra parte, la figura 3.34 muestra el mismo experimento cuando la tensión de entrada es 120 V, tensión de salida 300 V y potencia igual a 176 W. La tabla 3.8 muestra los resultados numéricos de los experimentos.

Como ya se indicó anteriormente, el sistema no mide la tensión de entrada y, por tanto, no puede detectar su distorsión. Este factor es, por tanto, crítico y es uno de los mayores inconvenientes de la técnica presentada.

3.6.6. Efectos de la resolución y cuantización del ADC en la corrección de factor de potencia

En el apartado 3.5 se vio la resolución del ADC y del regulador para evitar el ciclo límite. Sin embargo, no se analizó la influencia de la resolución del ADC en la corrección de factor de potencia. Este apartado va a mostrar, mediante resultados experimentales, cómo influye la resolución del ADC en el factor de potencia y en la distorsión armónica.

Dado que el sistema únicamente cuenta con un ADC para la regulación, es importante saber qué sensibilidad tiene el sistema frente a la resolución del ADC. Para comprobar esta sensibilidad, se han realizado nuevos experimentos. Se ha modificado el valor proporcionado por el ADC para simular un error de cuantización en el mismo. La variación del factor de potencia o de la distorsión armónica es muy baja cuando se modifica en 1 el valor del ADC. Por ello, se han hecho modificaciones mayores. En particular, se ha simulado un error igual a 16 LSB en la tensión media de salida, y se ha observado que el THD de corriente se modifica un 0,9879 %, por lo que cada LSB modifica de media un 0,0617 %. Por otra parte, se ha simulado el mismo error de 16 LSB en el rizado de la tensión de salida, y se ha comprobado que el THD de corriente cambia un 14,954 %, correspondiendo cada LSB a un 0,9346 %.

Los resultados, tomados con el equipamiento 1, muestran que el sistema es mucho más sensible ante errores de cuantización en la medida del rizado. Esto es normal,

ya que el mismo cambio de 1 LSB es mucho más representativo en el rizado de la tensión, el cual está en torno a 30 V, que en la tensión media, la cual está en torno a 400 V. Debido a esta circunstancia, la resolución del ADC puede llegar a ser crítica en el caso de que el rizado sea muy bajo, como puede ser cuando hay baja carga o una capacidad muy alta en el condensador de salida.

3.7. Conclusiones

En este capítulo se han mostrado varias técnicas para corrección de factor de potencia en las que se usan ciclos de trabajo precalculados. El uso del precálculo permite reducir el número de sensores, lo cual se traduce en menor tamaño y coste, entre otros factores. En contraposición a las técnicas clásicas de PFC, el sistema propuesto prescinde de ADCs para medir la tensión de entrada, y la corriente de entrada. Es significativo quitar el sensado de la corriente de entrada, pues suele ser más complejo, provocando pérdidas si se usa un *shunt*, y tampoco siendo óptimas las diferentes aproximaciones que existen, como un sensor de efecto *Hall*, que sube mucho el precio del sensor.

El sistema se basa en la lectura de los ciclos de trabajo precalculados y su aplicación en el momento adecuado. Para ello, el convertidor posee un comparador de tensión para la sincronización con la red eléctrica. Dado que los ciclos de trabajo han sido calculados para condiciones nominales, la corrección de factor de potencia se verá rápidamente afectada por condiciones de trabajo diferentes. Por ello, el capítulo ha mostrado cómo modificar el ciclo de trabajo precalculado para hacer frente a dichos cambios utilizando únicamente un ADC para medir la tensión de salida. En particular, s**e han descrito tres regulaciones diferentes** en orden creciente de prestaciones y de complejidad. Los tres métodos usan hasta dos lazos de control, los cuales hacen uso del **único ADC** que posee el sistema. El primer lazo utiliza la **tensión de salida media**, y se comporta como el **lazo de tensión** de las técnicas clásicas de PFC, midiendo el valor medio de la tensión de salida. Por su parte, el segundo lazo utiliza el **rizado de la tensión de salida**, que es proporcional al valor de la carga, comportándose como el **lazo de corriente** en las técnicas PFC habituales. Sin embargo, este último lazo tiene un ancho de banda mucho menor que el usado en técnicas clásicas y, de hecho, sólo actúa una vez por semiciclo de red.

Entre los tres regulaciones propuestas, el tercer método, d_a, d_b, d_c, obtiene los mejores resultados. En particular, se consigue un factor de potencia superior a 0,99 en condiciones nominales, y se mantiene considerablemente alto incluso en condiciones no nominales. Por ejemplo, tanto al 50 % de carga como con la tensión de entrada

un 10 % mayor o menor de lo esperado, el factor de potencia se mantiene igual o por encima de 0,98. Por último, se ha probado que el tercer sistema cumple la normativa IEC-61000-3-2.

El capítulo también muestra la estabilidad del sistema propuesto cuando se enfrenta a cambios en la bobina y condensador de salida frente a los valores nominales. Esto puede ocurrir por diferencias entre los valores de C y L que se usaron en el precálculo y los reales en el convertidor. Los resultados demuestran que en ambos casos el THD de corriente de entrada aumenta menos del 1 % cuando hay un error del 1 % en los parámetros L y C. Entre los resultados también se han cuantificado los efectos producidos por una tensión de entrada distorsionada. Se ha comprobado que el sistema propuesto es altamente sensible a dicha distorsión, siendo más sensible para la conversión de 230 V a 400 V, y siendo más robusto ante la conversión 120 V a 300 V. La alta sensibilidad ante tensiones distorsionadas es previsible, ya que el sistema no mide la tensión de entrada para ahorrar costes de fabricación, por lo que no actúa para evitarla.

Capítulo 4

Conclusiones

Este trabajo contiene dos capítulos principales, de temática relacionada pero muy dispar. Este capítulo de conclusiones resumirá las aportaciones principales de ambos temas.

4.1. Resumen de aportaciones sobre verificación de controladores digitales

La importancia de la simulación de los reguladores digitales antes de su prueba con sistemas reales está fuera de duda. No sólo se ha visto que es importante la simulación de un modelo del regulador, sino que es importante probar su implementación final para poder tener certeza sobre la corrección del diseño. El capítulo 2 ha propuesto modelar las plantas analógicas y simularlas junto al regulador en su implementación final. Una opción es utilizar un simulador mixto analógico-digital para poder simular la parte de potencia y la parte digital de forma conjunta. El gran inconveniente es que las simulaciones pueden ser excesivamente largas, especialmente en aplicaciones donde es necesario simular transitorios con dinámica lenta, como puede ser en corrección de factor de potencia. En cambio, el capítulo se ha centrado en modelar la planta en un lenguaje de descripción de *hardware*, para conseguir un entorno íntegramente digital, y mucho más rápido de simular. El capítulo ha mostrado diferentes aritméticas para modelar la citada planta en digital. Se ha explicado cómo hacer modelos usando aritméticas en coma flotante simulables (*real*), simulables y emulables (*float*) y en coma fija (QX.Y). Se ha visto que la primera es de gran facilidad de uso pero sólo permite su simulación, no optimizándose el tiempo de la etapa de pruebas. El tipo de datos *float* permite la emulación pero su simulación es extremadamente lenta, por

lo que pierde versatilidad. Por último el tipo de datos QX.Y es más complejo, ya que requiere que el diseñador tenga en cuenta el ancho de cada señal del modelo, pero este tipo permite la optimización máxima en recursos y tiempo de pruebas permitiendo, además, la emulación.

Todos los modelos se han comparado entre sí, y a la comparación se ha añadido la opción tradicional de simulación mixta analógica-digital y resultados experimentales para comprobar la validez de la propuesta presentada. Se ha comprobado que la simulación usando un modelo digital de la planta arroja resultados similares a los que se encontrarán en un convertidor real. Existen ciertas diferencias debido a las pérdidas eléctricas pero el comportamiento es muy similar, como se ha demostrado, haciendo que los resultados de las simulaciones y emulaciones sean muy significativos y útiles para las etapas de pruebas. Por otra parte, se ha mostrado cómo añadir pérdidas eléctricas de primer orden al modelo de la planta para hacer más realistas a las simulaciones. La adición de pérdidas no supone un esfuerzo excesivo y mejora ligeramente la simulación de los modelos.

Por último se ha realizado un análisis de resolución exhaustivo de las variables de estado del modelo de la planta. Las variables de estado deben guardar los valores de las tensiones y corrientes características del convertidor pero, a su vez, deben almacenar los pequeños incrementos que se producen en cada ciclo de integración. De hecho, cuanto más precisa se requiere la simulación, menor tiene que ser el tiempo de integración, y por tanto los incrementos serán más pequeños, haciendo que los anchos de las señales sean más grandes. Por ello, se requiere una gran cantidad de bits para almacenar las variables de estado. En el capítulo se ha visto que un tipo de datos de coma flotante de 32 bits no es suficiente para la simulación propuesta en el capítulo. Por tanto, se ha mostrado un análisis teórico y práctico sobre la resolución que deben tener las variables de estado en los modelos digitales.

Las principales aportaciones sobre verificación de controladores digitales son:

- Se han desarrollado diferentes entornos, basados en codificación HDL, de depuración de reguladores digitales que permiten tanto simulación como emulación en algunos casos. Los entornos utilizan las siguientes aritméticas: coma flotante *real* que sólo puede ser simulada, coma flotante *float* que también puede ser emulada, coma fija (QX.Y), y coma fija con la biblioteca *sfixed*, siendo las dos últimas simulables y emulables.
- Se ha realizado una comparativa de los diferentes métodos teniendo en cuenta criterios como velocidad de simulación y de emulación, recursos hardware y facilidad de diseño.

- Se han comparado las propuestas con la simulación mixta analógica-digital, que es el método tradicional de depuración, y con resultados experimentales, demostrando la validez de las propuestas presentadas durante el capítulo.
- Se ha añadido pérdidas de primer orden a los modelos y se ha comparado la mejora de la precisión de la simulación.
- Se ha realizado un análisis de resolución. Hasta ahora se asumía que coma flotante de 32 bits tenía rango de representación numérica suficiente. Se ha demostrado que la aplicación presentada no puede simularse mediante coma flotante de 32 bits, y se ha analizado qué factores determinan la resolución. Se ha comprobado que la frecuencia de conmutación influye en gran medida sobre el ancho que deben tener las variables de estado del modelo.

4.2. Resumen de aportaciones sobre el uso de ciclo de trabajo precalculado para corrección de factor de potencia

Las técnicas clásicas de corrección de factor de potencia necesitan sensar tres variables analógicas: tensión de entrada, tensión de salida y corriente de entrada. En la presente tesis doctoral se ha presentado un método que realiza corrección de factor de potencia pero midiendo únicamente la tensión de salida, aparte de una detección simple del paso por cero de la tensión de entrada. La forma de conseguirlo es precalcular el ciclo de trabajo que debe aplicarse al interruptor del convertidor conmutado. Dado que la rectificación de la corriente alterna es una tarea periódica, se pueden calcular a priori los ciclos de trabajo correspondientes a un número limitado de ciclos de conmutación y posteriormente aplicarlos repetidamente. El capítulo muestra cómo calcular los ciclos de trabajo para unas ciertas condiciones, y muestra cómo aplicarlos al interruptor cuando es necesario. Para ello, los ciclos de trabajo han sido grabados en una memoria, y la sincronización necesaria con la red eléctrica se ha conseguido mediante un comparador analógico de tensión con la tensión de entrada rectificada.

Idealmente el sistema sólo necesitaría la citada memoria y el comparador para la sincronización de la misma con la red. Sin embargo, las condiciones reales de trabajo varían y se hace imprescindible realizar una regulación en tiempo real. Para ello se han presentado varios métodos que permiten realizar dicha regulación modificando los ciclos de trabajo precalculados. Las regulaciones presentadas usan un único ADC para medir la tensión de salida. Por una parte, se consigue la tensión media de salida, parámetro a regular en un corrector de factor de potencia. Los cambios que pueda

haber en la tensión de entrada se ven reflejados en la tensión media de salida y un lazo los corrige. Por otra parte, los cambios de la carga del convertidor empeoran el factor de potencia, por lo que el ciclo de trabajo requiere más regulaciones. Para ello se usa el mismo ADC para medir el rizado de la tensión de salida, ya que la carga del convertidor es proporcional al rizado de la tensión de salida. El sistema, por tanto, mide indirectamente la carga del convertidor y modifica el ciclo de trabajo de acuerdo a ella. Se han probado todos los métodos propuestos y todos cumplen la normativa IEC-61000-3-2, la cual regula la corrección de factor de potencia, midiendo el contenido armónico de la corriente de entrada. Asimismo, todos los métodos cumplen normativa ante variaciones notables en la tensión de entrada y en la carga conectada al convertidor. Como principal contrapartida de los métodos propuestos, al no medirse la tensión de entrada, el contenido armónico en la misma puede empeorar notablemente el factor de potencia.

Las principales aportaciones sobre el uso de ciclo de trabajo precalculado son:

- Se ha realizado corrección de factor de potencia midiendo únicamente la tensión de salida con un único ADC y un comparador de tensión para la sincronización con la red eléctrica.
- Se ha precalculado el ciclo de trabajo que debe tener el interruptor del convertidor, y se ha realizado un análisis del citado ciclo de trabajo en una, dos y tres componentes.
- Se ha estudiado la regulación de las distintas componentes descritas del ciclo de trabajo usando uno o dos lazos. El lazo principal regula la tensión media de salida, mientras que el segundo lazo se adapta a la carga del convertidor en función del rizado de la tensión de salida, dado que el rizado es proporcional a la carga del convertidor.

4.3. Trabajo futuro

La línea de investigación abierta no se cierra con la presentación de la tesis doctoral sino que abre nuevas líneas de trabajo. Las líneas futuras sobre la verificación de reguladores digitales son:

- Se extenderá la emulación HIL a otras aplicaciones, tales como control de motores, corriente trifásica, convertidores continua-continua, etc.

- Las técnicas de simulación propuestas a priori buscan la validación del regulador sacrificando cierta precisión en la simulación para favorecer el tiempo de simulación. Aprovechando la aceleración usando una arquitectura HIL, el capítulo mostró cómo añadir pérdidas eléctricas de primer orden a la planta. Se propone hacer un modelo aún más realista, ya que la emulación aún así será suficientemente rápida.

Las líneas futuras sobre la aplicación de ciclo precalculado para corrección de factor de potencia son:

- Se estudiará la posibilidad de añadir un tercer lazo para corregir efectos como la sincronización con red, retardos en la conmutación, etc, pero siempre midiendo la tensión de salida para no aumentar el coste del regulador. Estos errores influyen en la tensión de salida, haciendo que su punto máximo y mínimo estén en instantes diferentes al ideal. Se propone adelantar o retrasar la aplicación de los valores precalculados, para que las crestas de v_{out} estén en el instante de tiempo $\frac{1}{4}T_{Sw} = 2{,}5\ ms$ y los valles estén en en el instante de tiempo $\frac{3}{4}TSw = 7{,}5\ ms$.
- Como se ha visto, el sistema precalculado es muy vulnerable ante tensión de entrada distorsionada. Siguiendo la filosofía de medir únicamente la tensión de salida, se estudiará la posibilidad de añadir un módulo de cálculo de la FFT de la tensión de salida, ya que dicha tensión de salida se ve también distorsionada cuando la tensión de entrada está distorsionada. Se estudiará la utilidad de esta información para adaptar el ciclo de trabajo.

4.4. Publicaciones derivadas de la tesis doctoral

4.4.1. Relacionadas con la verificación de controladores digitales

- **Alberto Sanchez**, Angel de Castro y Javier Garrido: *A Comparison of Simulation and Hardware-in-the-Loop Alternatives for Digital Control of Power Converters*, en: Industrial Informatics, IEEE Transactions on, volumen 8, nº3 págs. 491-500, 2012. Revista con índice de impacto 2,990 y Q1 en el JCR (año 2011).
- **Alberto Sanchez**, Angel de Castro y Javier Garrido, *Real-Time Hardware-in-the-Loop Emulation for Boost Power Factor Corrector*, en: XXVI Conference on Design of Circuits and Integrated Systems (DCIS), págs. 363-368 (2011).

- Fernando López Colino, **Alberto Sanchez**, Angel de Castro y Javier Garrido, *Modeling of Power Converters for Debugging Digital Controllers through FPGA Emulation*, en: XV European Conference on Power Electronics and Applications (EPE ECCE), 2013. Aceptado. El congreso se celebrará en septiembre de 2013.
- Fernando López Colino, **Alberto Sanchez**, Angel de Castro y Javier Garrido, *Depuración de reguladores digitales para convertidores conmutados utilizando emulación en FPGA*, en: XX Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI), 2013. Aceptado. El congreso se celebrará en julio de 2013.

4.4.2. Relacionadas con la aplicación de ciclo de trabajo precalculado para corrección de factor de potencia

- **Alberto Sanchez**, Angel de Castro, Victor M. López, Francisco J. Azcondo y Javier Garrido: *Single ADC Digital PFC Controller using Pre-calculated Duty Cycles*, en: Power Electronics, IEEE Transactions on. Aceptado. En prensa. Revista con índice de impacto 4,650 y Q1 en el JCR (año 2011).
- **Alberto Sanchez**, Angel de Castro y Javier Garrido, *Single ADC Single Loop Power Factor Correction using Pre-Calculated Duty Cycles*, en: XV European Conference on Power Electronics and Applications (EPE ECCE), 2013. Aceptado. El congreso se celebrará en septiembre de 2013.
- **Alberto Sanchez**, Fernando López Colino, Angel de Castro y Javier Garrido, *Corrección de factor de potencia en lazo simple utilizando valores precalculados para el ciclo de trabajo*, en: XX Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI), 2013. Aceptado. El congreso se celebrará en julio de 2013.

Apéndice A

Listado de códigos

A.1. Modelos del convertidor elevador para simulación o emulación

A.1.1. Modelo *real*

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;

entity BoostConverterReal is
port (
        -- Input ports
        Clk : in std_logic;
        Reset : in std_logic;
        Mosfet : in std_logic; -- On = '1', off = '0'
        Vg : in real;
        Ir : in real;
        -- Output ports
        Iin : out real;
        Vout : out real
);
end BoostConverterReal;

architecture Behavioral of BoostConverterReal is
        constant C : real := 0.0001;
        constant L : real := 0.005;
        constant dt : real := 0.00000001;

        signal iL : real := 0.0;
        signal voutAux : real := 400.0;
        signal iLAdd, voutAuxAdd : real := 0.0;
```

```
        constant VOINIT : real := 400.0;
        constant ILINIT : real := 0.0;

        constant dtL : real := dt/L;
        constant dtC : real := dt/C;
begin
        Iin <= iL;
        Vout <= voutAux;

        SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
        -- Selection (multiplexer) of values to be added to input current and
        -- output voltage
        begin
                if Mosfet = '1' then -- Closed switch
                        iLAdd <= Vg;
                        voutAuxAdd <= -(Ir);
                else -- Open switch
                        if iL > 0.0 then -- CCM
                                iLAdd <= (Vg - voutAux);
                                voutAuxAdd <= (iL - Ir);
                        else -- DCM
                                iLAdd <= 0.0;
                                voutAuxAdd <= -(Ir);
                        end if;
                end if;
        end process SWITCHMUX;

        DIFFEQ: process(Clk, Reset)
        -- Update of Vout and Iin each clock cycle
        begin
                if Reset = '1' then
                        voutAux <= VOINIT;
                        iL <= ILINIT;
                elsif rising_edge(Clk) then
                        iL <= iL + iLAdd*dtL;
                        voutAux <= voutAux + voutAuxAdd*dtC;
                end if;
        end process DIFFEQ;
end Behavioral;
```

Código A.1: Modelo *real* del convertidor elevador

A.1.2. Modelo *float*

```
library IEEE;
use IEEE.std_logic_1164.all;

library ieee_proposed;
use ieee_proposed.float_pkg.all;
use ieee_proposed.fixed_pkg.all;
use ieee_proposed.fixed_float_types.all;
```

```
entity BoostConverterFloat is
port(
        -- Input ports
        Clk : in std_logic;
        Reset : in std_logic;
        Mosfet : in std_logic; -- On = '1', off = '0'
        Vg : in float32;
        Ir : in float32;
        -- Output ports
        Iin : out float32;
        Vout : out float32
);
end BoostConverterFloat;

architecture Behavioral of BoostConverterFloat is

        constant C : float32 := to_float(0.0001);
        constant L : float32 := to_float(0.005);
        constant dt : float32 := to_float(0.00000001);

        signal iL, voutAux : float32;
        signal iLAdd : float32;
        signal voutAuxAdd : float32;

        constant VOINIT : float32 := to_float(400.0);
        constant ILINIT : float32 := to_float(0.0);
        constant CZERO : float32 := to_float(0.0);

        constant dtL : float32 := dt/L;
        constant dtC : float32 := dt/C;

        -------Debug signals (type real)------------
        -- Comment the following line before synthesizing
        -- signal voutReal, iinReal : real;
        -- signal voutAuxAddReal, iLAddReal : real;
        -------Debug signals (type real)------------

begin

        Iin <= iL;
        Vout <= voutAux;

        ---------Debug signals (type real)----------
        -- Comment from here before synthesizing
        -- VoutReal <= to_real(voutAux);
        -- IinReal <= to_real(iL);
        -- voutAuxAddReal <= to_real(voutAuxAdd);
        -- iLAddReal <= to_real(iLAdd);
        -- Comment up to here before synthesizing
        --------------SEÑALES DE DEBUG--------------

        SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
        -- Selection (multiplexer) of values to be added to input
        -- current and output voltage
        begin
```

```
                if Mosfet = '1' then -- Closed switch
                        iLAdd <= Vg;
                        voutAuxAdd <= -(Ir);
                else -- Open switch
                        if gt(iL, CZERO) then -- CCM (gt: greater than)
                                iLAdd <= (Vg - voutAux);
                                voutAuxAdd <= (iL - Ir);
                        else -- DCM
                                iLAdd <= CZERO;
                                voutAuxAdd <= -(Ir);
                        end if;
                end if;
        end process SWITCHMUX;

        DIFFEQ: process(Clk, Reset)
        -- Update of Vout and Iin each clock cycle
        begin
                if Reset = '1' then
                        voutAux <= VOINIT;
                        iL <= ILINIT;
                elsif rising_edge(Clk) then
                        iL <= iL + iLAdd*dtL;
                        voutAux <= voutAux + voutAuxAdd*dtC;
                end if;
        end process DIFFEQ;

end Behavioral;
```

Código A.2: Modelo *float* del convertidor elevador

A.1.3. Modelo en coma fija

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;
use IEEE.numeric_std.all;

entity BoostConverterQXY is
port(
        -- Input ports
        Clk : in std_logic;
        Reset : in std_logic;
        Mosfet : in std_logic; -- On = '1', off = '0'
        Vg : in std_logic_vector(12 downto 0); -- Q9.3 V
        Ir : in std_logic_vector(12 downto 0); -- Q22.-10 dt/L A
        -- Output ports
        Iin : out std_logic_vector(11 downto 0); -- Q3.9 A w/o sign bit
        Vout : out std_logic_vector(11 downto 0) -- Q10.2 V w/o sign bit
);
end BoostConverterQXY;

architecture Behavioral of BoostConverterQXY is
```

```
  -- 400.0 V: Init value of voutAux in Q43.-10 dt/C dt/L
        constant VOAUXINIT : std_logic_vector(33 downto 0) :=
                "0001110100011010100101001010001000";
        -- Init value of iL, 0 A
        constant ILINIT : std_logic_vector(25 downto 0)
                := (others => '0');
        -- dt/C dt/L in Q-32.49
        constant VOUTSCALE : std_logic_vector(17 downto 0)
                := "011011011111001110";
        -- dt/L in Q-18.35
        constant IINSCALE : std_logic_vector(17 downto 0)
                := "010000110001101111";

        -- Format Q22.3 dt/L
        signal iL : std_logic_vector(25 downto 0);
        -- Format Q22.-5 dt/L
        signal iLSat : std_logic_vector(17 downto 0);
        -- Format Q43.-10 dt/L dt/C
        signal voutAux : std_logic_vector(33 downto 0);
        -- Format Q43.-26 dt/L dt/C
        signal voutAuxSat : std_logic_vector(17 downto 0);
        -- Vout in Q12.23, without dt/L dt/C after being scaled
        signal voutScaled : std_logic_vector(35 downto 0);
        -- Il in Q5.30, without dt/L after being scaled
        signal iinScaled : std_logic_vector(35 downto 0);
        -- Value to be added to iL. Format Q9.3
        signal iLAdd : std_logic_vector (12 downto 0);
        -- Value to be added to voutAux. Format Q22.-10
        signal voutAuxAdd : std_logic_vector (12 downto 0);
        -- Vout in Q9.3. Used to feedback the model. Max value is 512 V
        signal voutFeedback     : std_logic_vector(12 downto 0);

        -------Debug signals (type real)------------
        -- Comment the following line before synthesizing
        signal voutReal, iinReal        : real;
        ------------------------------------------------

begin

        ---------Debug signals (type real)----------
        -- Comment from here before synthesizing
        conv_integer accepts only 32 bits
        voutReal <= real(conv_integer(voutScaled(35 downto 4)))/(2.0**19);
        conv_integer accepts only 32 bits
        iinReal <= real(conv_integer(iinScaled(35 downto 4)))/(2.0**26);
        -- Comment up to here before synthesizing
        -------Debug signals (type real)------------


        -- Internal voutAux and iL are scaled to volts and amperes

        --Q43.-26 dt/C dt/L.
        -- Truncated in order to fit in a 18x18 multiplier
        voutAuxSat <= voutAux(33 downto 16);
```

```
        -- Q12.23 = Q43.-26 * Q-32.49
        voutScaled <= voutAuxSat * VOUTSCALE;
        -- To be added with Vg. Q9.3.
        -- If vout > 512 V, voutFeedback overflows.
        voutFeedback <= voutScaled(32 downto 20);
        -- Q10.2 without sign bit
        Vout <= voutScaled(32 downto 21) when voutScaled(32) = '0'
                else (others => '0');

        -- Q22.-5 dt/L. Truncated in order to fit in a 18x18 multiplier
        iLSat <= iL(25 downto 8);
        -- Q5.30 = Q22.-5 * Q-18.35
        iinScaled <= iLSat * IINSCALE;
        -- Q3.9 without sign bit
        Iin <= iinScaled(32 downto 21) when iinScaled(32) = '0'
                else (others => '0');

        SWITCHMUX : process (Mosfet, Vg, Ir, iL, voutFeedback)
        begin
                if Mosfet = '1' then -- Closed switch
                        iLAdd <= Vg;
                        voutAuxAdd <= -Ir;
                else -- Open switch
                        if iL > conv_std_logic_vector(0, iL'length) then -- CCM
                                iLAdd <= Vg - voutFeedback;
                                -- iL is truncated to the same scale than Ir
                                voutAuxAdd <= iL(25 downto 13) - Ir;
                        else -- DCM
                                iLAdd <= (others => '0');
                                voutAuxAdd <= - Ir;
                        end if;
                end if;
        end process SWITCHMUX;

        DIFFEQ: process(Reset, Clk)
        -- Update of Vout and Iin each clock cycle
        begin
                if Reset = '1' then
                        voutAux <= VOAUXINIT;
                        iL <= ILINIT;
                elsif rising_edge(Clk) then
                        iL <= iL + iLAdd;
                        voutAux <= voutAux + voutAuxAdd;
                end if;
        end process DIFFEQ;

end Behavioral;
```

Código A.3: Modelo en coma fija del convertidor elevador

A.1.4. Modelo en coma fija usando la biblioteca *sfixed*

```
library IEEE, ieee_proposed;
```

```
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;

entity BoostConverterQXY is
        generic (
                DT : real := 10.0e-9;
                C : real := 0.000100;
                L : real := 0.005;
                VOINIT : real := 400.0;
                IOINIT : real := 0.0;
        );
        port(
                -- Input ports
                Clk : in std_logic;
                Reset : in std_logic;
                Mosfet : in std_logic; -- On = '1', off = '0'
                Vg : in std_logic_vector(12 downto 0); -- Q9.3 V
                Ir : in std_logic_vector(12 downto 0); -- Q22.-10 DT/L A
                -- Output ports
                Iin : out std_logic_vector(11 downto 0); -- Q3.9 A w/o sign bit
                Vout : out std_logic_vector(11 downto 0) -- Q10.2 V w/o sign bit
        );
end BoostConverterQXY;

architecture Behavioral of BoostConverterQXY is

        -- A QX.Y number is represented using a fixed(X downto -Y)

        -- VOINIT V: Init value of voutAux in Q43.-10 DT/C DT/L.
        constant VOAUXINIT : sfixed(43 downto 10) := to_sfixed((VOINIT*C*L)/(DT*DT),
                43,10);
        -- 0.0 A: Init value of iL in Q22.3
        constant ILINIT : sfixed(22 downto -3) := to_sfixed(IOINIT,22,-3);

        -- DT/C DT/L in Q-32.49
        constant VOUTSCALE : sfixed(-32 downto -49) := "0110110111111001110";
        -- DT/L in Q-15.35
        constant IINSCALE : sfixed(-15 downto -35) := to_sfixed(DT/L,-15,-35);

        -- Format Q22.3 DT/L
        signal iL : sfixed(22 downto -3);
        -- Format Q22.-5 DT/L
        signal iLSat : sfixed(22 downto 5);
        -- Format Q43.-10 DT/L DT/C
        signal voutAux : sfixed(43 downto 10);
        -- Format Q43.-26 DT/L DT/C
        signal voutAuxSat : sfixed(43 downto 26);
        -- Vout in Q12.23, without DT/L DT/C after being scaled
        signal voutScaled : sfixed(12 downto -23);
        -- Il in Q5.30, without DT/L after being scaled
        signal iinScaled : sfixed(5 downto -30);
```

```
        -- Value to be added to iL. Format Q9.3
        signal iLAdd : sfixed(9 downto -3);
        -- Value to be added to voutAux. Format Q22.-10
        signal voutAuxAdd : sfixed(22 downto 10);
        -- Value of IR. Format Q22.-10. To be subtracted from voutAuxAdd
        signal vIr : sfixed(22 downto 10);
        --Value of VG. Format Q9.3
        signal vVg : sfixed(9 downto -3);
        -- Vout in Q9.3. Used to feedback the model. Max value is 512 V
        signal voutFeedback : sfixed(9 downto -3);

        signal voutaux2 : ufixed (9 downto -2);
        signal iIn2 : ufixed (2 downto -9);

        -------Debug signals (type real)------------
        signal voutAuxReal, iLReal      : real;
        signal voutScaledReal, iinScaledReal    : real;
        signal iLAddReal, voutAuxAddReal : real;
        signal voutADCReal, iinADCReal : real;
        signal iInScaleReal, voutScaleReal : real;
        -------Debug signals (type real)------------

begin

        -------Debug signals (type real)------------
        voutScaledReal <= to_real(voutScaled);
        iinScaledReal <= to_real(iinScaled);

        iInScaleReal <= to_real(IINSCALE);
        voutScaleReal <= to_real(VOUTSCALE);

        iLAddReal <= to_real(iLAdd)*DT/(L);
        voutAuxAddReal <= to_real(voutAuxAdd)*(dt/L)*(dt/C);

        voutAuxReal <= to_real(voutAux)*(dt/L)*(dt/C);
        iLReal <= to_real(iL)*dt/(L);

        iInADCReal <= to_real(iIn2);
        voutADCReal <= to_real(voutaux2);
        -------Debug signals (type real)------------

        -- Internal voutAux and iL are scaled to volts and amperes

        -- Truncated in order to fit in a 18x18 multiplier
        voutAuxSat <= resize(voutAux,voutAuxSat);
        -- Q12.23 = Q43.-26 * Q-32.49
        voutScaled <= resize(voutAuxSat * VOUTSCALE, voutScaled);
        -- To be added with Vg. Q9.3. If vout > 512 V, voutFeedback overflows.
        voutFeedback <= resize(voutScaled,voutFeedback);
        -- Q10.2 without sign bit
        voutaux2 <= resize(ufixed(voutScaled),voutaux2) when voutScaled >= 0 else
                (others => '0');
        Vout <= to_slv(voutaux2);

        -- Q22.-5 DT/L. Truncated in order to fit in a 18x18 multiplier
```

```
        iLSat <= resize(iL,iLSat);
        -- Q5.30 = Q22.-5 * Q-18.35
        iinScaled <= resize(iLSat * IINSCALE,iinScaled);
        -- Q3.9 without sign bit
        Iin2 <=  resize(ufixed(iinScaled),Iin2) when iinScaled >= 0
                else (others => '0');
        Iin <= to_slv(Iin2);

        vVg <= to_sfixed (Vg,vVG);
        vIr <= to_sfixed(Ir,voutAuxAdd);

        SWITCHMUX : process (Mosfet, vVg, vIr, iL, voutFeedback)
        begin
                if Mosfet = '1' then -- Closed switch
                        iLAdd <= resize(vVg, iLAdd);
                        voutAuxAdd <= resize(- vIr, voutAuxAdd);
                else -- Open switch
                        if iL > 0 then -- CCM
                                iLAdd <= resize(vVg - voutFeedback,iLAdd);
                                voutAuxAdd <= resize ( resize(iL, voutAuxAdd)
                                        - vIr, voutAuxAdd);
                        else -- DCM
                                iLAdd <= to_sfixed(0.0, iLAdd);
                                voutAuxAdd <= resize(-vIr, voutAuxAdd);
                        end if;
                end if;
        end process SWITCHMUX;

        DIFFEQ: process(Reset, Clk)
        -- Update of Vout and Iin each clock cycle
        begin
                if Reset = '1' then
                        voutAux <= VOAUXINIT;
                        iL <= ILINIT;
                elsif rising_edge(Clk) then
                        iL <= resize(iL + iLAdd,iL);
                        voutAux <= resize(voutAux + voutAuxAdd,voutAux);
                end if;
        end process DIFFEQ;

end Behavioral;
```

Código A.4: Modelo en coma fija del convertidor elevador

A.1.5. Modelo en coma fija con pérdidas eléctricas

```
library IEEE, ieee_proposed;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;
```

```
entity BoostConverterQXY is
        generic (
                DT : real := 10.0e-9;
                C : real := 0.000100;
                L : real := 0.005;
                VOINIT : real := 400.0;
                IOINIT : real := 0.0;
                VB : real := 0.0; -- Forward voltage drop
                RM : real := 0.0; -- MOSFET ON resistance
                VD : real := 0.0; -- Forward voltage drop
                RL : real := 0.0 -- Series resistance of the inductor
        );
        port(
                -- Input ports
                Clk : in std_logic;
                Reset : in std_logic;
                Mosfet : in std_logic; -- On = '1', off = '0'
                Vg : in std_logic_vector(12 downto 0); -- Q9.3 V
                Ir : in std_logic_vector(12 downto 0); -- Q22.-10 DT/L A
                -- Output ports
                Iin : out std_logic_vector(11 downto 0); -- Q3.9 A w/o sign bit
                Vout : out std_logic_vector(11 downto 0) -- Q10.2 V w/o sign bit
        );
end BoostConverterQXY;

architecture Behavioral of BoostConverterQXY is

        -- A QX.Y number is represented using a fixed(X downto -Y)

        -- VOINIT V: Init value of voutAux in Q43.-10 DT/C DT/L.
        constant VOAUXINIT : sfixed(43 downto 10) := to_sfixed((VOINIT*C*L)/(DT*DT),
                43,10);
        -- 0.0 A: Init value of iL in Q22.3
        constant ILINIT : sfixed(22 downto -3) := to_sfixed(IOINIT,22,-3);

        -- DT/C DT/L in Q-32.49
        constant VOUTSCALE : sfixed(-32 downto -49) := "0110110111111001110";
        -- DT/L in Q-15.35
        constant IINSCALE : sfixed(-15 downto -35) := to_sfixed(DT/L,-15,-35);

        -- Format Q22.3 DT/L
        signal iL : sfixed(22 downto -3);
        -- Format Q22.-5 DT/L
        signal iLSat : sfixed(22 downto 5);
        -- Format Q43.-10 DT/L DT/C
        signal voutAux : sfixed(43 downto 10);
        -- Format Q43.-26 DT/L DT/C
        signal voutAuxSat : sfixed(43 downto 26);
        -- Vout in Q12.23, without DT/L DT/C after being scaled
        signal voutScaled : sfixed(12 downto -23);
        -- Il in Q5.30, without DT/L after being scaled
        signal iinScaled : sfixed(5 downto -30);
        -- Value to be added to iL. Format Q9.3
        signal iLAdd : sfixed(9 downto -3);
```

```
        -- Value to be added to voutAux. Format Q22.-10
        signal voutAuxAdd : sfixed(22 downto 10);
        -- Value of IR. Format Q22.-10. To be subtracted from voutAuxAdd
        signal vIr : sfixed(22 downto 10);
        --Value of VG. Format Q9.3
        signal vVg : sfixed(9 downto -3);
        -- Vout in Q9.3. Used to feedback the model. Max value is 512 V
        signal voutFeedback : sfixed(9 downto -3);

        signal voutaux2 : ufixed (9 downto -2);
        signal iIn2 : ufixed (2 downto -9);


        -- Signals for electrical looses --
        signal vRMI1 : sfixed(9 downto -3); --Value of VD scaled
        signal vRLI1 : sfixed(9 downto -3); --Value of VD scaled
        signal vVB : sfixed(9 downto -3); --Value of VB using fixed point.
        signal vVgIn : sfixed(9 downto -3); --Value of VG corrected.
        signal vVG : sfixed(9 downto -3); --Value of VG .
        signal vVD : sfixed(9 downto -3); --Value of VD .

        -------Debug signals (type real)------------
        signal voutAuxReal, iLReal      : real;
        signal voutScaledReal, iinScaledReal    : real;
        signal iLAddReal, voutAuxAddReal : real;
        signal voutADCReal, iinADCReal : real;
        signal iInScaleReal, voutScaleReal : real;
        -------Debug signals (type real)------------

begin

        -------Debug signals (type real)------------
        voutScaledReal <= to_real(voutScaled);
        iinScaledReal <= to_real(iinScaled);

        iInScaleReal <= to_real(IINSCALE);
        voutScaleReal <= to_real(VOUTSCALE);

        iLAddReal <= to_real(iLAdd)*DT/(L);
        voutAuxAddReal <= to_real(voutAuxAdd)*(dt/L)*(dt/C);

        voutAuxReal <= to_real(voutAux)*(dt/L)*(dt/C);
        iLReal <= to_real(iL)*dt/(L);

        iInADCReal <= to_real(iIn2);
        voutADCReal <= to_real(voutaux2);
        -------Debug signals (type real)------------

        -- Internal voutAux and iL are scaled to volts and amperes

        -- Truncated in order to fit in a 18x18 multiplier
        voutAuxSat <= resize(voutAux,voutAuxSat);
        -- Q12.23 = Q43.-26 * Q-32.49
        voutScaled <= resize(voutAuxSat * VOUTSCALE, voutScaled);
        -- To be added with Vg. Q9.3. If vout > 512 V, voutFeedback overflows.
```

```
voutFeedback <= resize(voutScaled,voutFeedback);
-- Q10.2 without sign bit
voutaux2 <= resize(ufixed(voutScaled),voutaux2) when voutScaled >= 0
        else (others => '0');
Vout <= to_slv(voutaux2);

-- Q22.-5 DT/L. Truncated in order to fit in a 18x18 multiplier
iLSat <= resize(iL,iLSat);
-- Q5.30 = Q22.-5 * Q-18.35
iinScaled <= resize(iLSat * IINSCALE,iinScaled);
-- Q3.9 without sign bit
Iin2 <=  resize(ufixed(iinScaled),Iin2) when iinScaled >= 0
        else (others => '0');
Iin <= to_slv(Iin2);

-- Electrical looses
vRLI1 <= resize (RL * iL * IINSCALE , vRLI1);
vRMI1 <= resize (RM * iL * IINSCALE , vRMI1);
vVD <= to_sfixed (VD , vVD);
vVB <= to_sfixed (VB, vVB);

vVg <= to_sfixed (Vg,vVG);
vVgIn <= resize((vVG - vVB),vVgIn) when (vVG > vVB) else (others => '0');
vIr <= to_sfixed(Ir,voutAuxAdd);

SWITCHMUX : process (Mosfet, vVgIn, vIr, iL, voutFeedback, vRLI1, vRMI1, vVD)
begin
        if Mosfet = '1' then -- Closed switch
                iLAdd <= resize(vVgIn - vRLI1 - vRMI1, iLAdd);
                voutAuxAdd <= resize(- vIr, voutAuxAdd);
        else -- Open switch
                if iL > 0 then -- CCM
                        iLAdd <= resize(vVgIn - voutFeedback - vVD - vRLI1,
                                iLAdd);
                        voutAuxAdd <= resize ( resize(iL, voutAuxAdd) - vIr,
                                voutAuxAdd);
                else -- DCM
                        iLAdd <= to_sfixed(0.0, iLAdd);
                        voutAuxAdd <= resize(-vIr, voutAuxAdd);
                end if;
        end if;
end process SWITCHMUX;



DIFFEQ: process(Reset, Clk)
-- Update of Vout and Iin each clock cycle
begin
        if Reset = '1' then
                voutAux <= VOAUXINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then
                iL <= resize(iL + iLAdd,iL);
                voutAux <= resize(voutAux + voutAuxAdd,voutAux);
        end if;
```

```
        end process DIFFEQ;

end Behavioral;
```

Código A.5: Modelo en coma fija del convertidor elevador con pérdidas eléctricas

A.1.6. Modelo *real* con pérdidas eléctricas

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;

entity BoostConverterReal is
        generic (
                C : real := 0.000068;
                L : real := 0.005;
                VOINIT : real := 400.0;
                VB : real := 0.0; -- Forward voltage drop
                RM : real := 0.0; -- MOSFET ON resistance
                VD : real := 0.0; -- Forward voltage drop
                RL : real := 0.0 -- Series resistance of the inductor
        );
        port (
                --In
                Clk : in std_logic;
                Reset : in std_logic;
                Mosfet : in std_logic; -- On = '1', off = '0'
                Vg : in real;
                Ir : in real;
                -- Out
                Iin : out real;
                Vout : out real
        );
end BoostConverterReal;

architecture Behavioral of BoostConverterReal is

        constant dt : real := 0.00000001;

        signal iL : real := 0.0;
        signal voutAux : real := VOINIT;
        signal iLAdd, voutAuxAdd : real := 0.0;

        constant ILINIT : real := 0.0;
        constant dtL : real := dt/L;
        constant dtC : real := dt/C;

        signal VgReal : real;


begin
```

```
        Iin <= iL;
        Vout <= voutAux;

        VgReal <= Vg - VB when Vg > VB else 0.0;

        SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
        begin
                if Mosfet = '1' then -- Closed sswitch
                        iLAdd <= VgReal - (RL*Il) - (RM * Il);
                        voutAuxAdd <= -(Ir);
                else -- Open switch
                        if iL > 0.0 then -- CCM
                                iLAdd <= (VgReal - voutAux - Vd) - (RL*Il);
                                voutAuxAdd <= (iL - Ir);
                        else -- DCM
                                iLAdd <= 0.0;
                                voutAuxAdd <= -(Ir);
                        end if;
                end if;
        end process SWITCHMUX;

  DIFFEQ: process(Clk, Reset)
        -- Update of Vout and Iin each clock cycle
        begin
                if Reset = '1' then
                        voutAux <= VOINIT;
                        iL <= ILINIT;
                elsif rising_edge(Clk) then
                        if Vg<1.0 then
                                iL <= 0.0;
                        else
                                iL <= iL + iLAdd*dtL;
                        end if;
                        voutAux <= voutAux + voutAuxAdd*dtC;
                end if;
        end process DIFFEQ;

end Behavioral;
```

Código A.6: Modelo *real* del convertidor elevador con pérdidas eléctricas

A.2. Modelos del ADC

A.2.1. Modelo del ADC para simulación en *real*

```
library IEEE, WORK;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;
```

```
use WORK.MyConvPack.all; -- Function floor

entity AdcReal2StdVector is
generic(
        CYCLESDELAY : integer := 5;
        NBITS : integer := 12
);
port(
        Clk : in std_logic;
        Reset : in std_logic;
        Start : in std_logic;
        AnalogIn : in real;
        DataOut : out std_logic_vector(NBITS-1 downto 0)
);
end AdcReal2StdVector;

architecture Behavioral of AdcReal2StdVector is

        signal sample : real;
        signal counter : integer range 0 to CYCLESDELAY;
        signal startR, startR2 : std_logic;

begin

        prStart : process(Reset, Clk)
        begin
                if Reset = '1' then
                        startR <= '0';
                        startR2 <= '0';
                elsif rising_edge(Clk) then
                        startR <= Start;
                        startR2 <= StartR;
                end if;
        end process prStart;

        prDelay : process(Reset, Clk)
        begin
                if Reset = '1' then
                        counter <= 0;
                        sample <= 0.0;
                        DataOut <= (others => '0');
                elsif rising_edge(Clk) then
                        if StartR = '1' and startR2 = '0' then
                                counter <= 1;
                                if (AnalogIn < 0.0) then
                                        sample <= 0.0;
                                elsif (AnalogIn >= real(2**NBITS-1)) then
                                        sample <= real(2**NBITS-1);
                                else
                                        sample <= AnalogIn;
                                end if;
                        elsif counter = CYCLESDELAY then
                                DataOut <= conv_std_logic_vector
                                                (floor(sample), NBITS);
                        elsif counter /= 0 then
```

```
                                counter <= counter + 1;
                        end if;
                end if;
        end process prDelay;

end Behavioral;
```

Código A.7: Modelo del ADC para simulación en *real*

A.2.2. Modelo del ADC para simulación/emulación en *float*

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

library ieee_proposed;
use ieee_proposed.float_pkg.all;
use ieee_proposed.fixed_pkg.all;
use ieee_proposed.fixed_float_types.all;

entity AdcFloat2StdVector is
generic(
        CYCLESDELAY : integer := 5;
        NBITS : integer := 12
);
port(
        Clk : in std_logic;
        Reset : in std_logic;
        Start : in std_logic;
        AnalogIn : in float32;
        DataOut : out std_logic_vector(NBITS-1 downto 0);
);
end AdcFloat2StdVector;

architecture Behavioral of AdcFloat2StdVector is

        signal sample : float32;
        signal counter : integer range 0 to CYCLESDELAY;
        signal startR, startR2 : std_logic;

begin

        prStart : process(Reset, Clk)
        begin
                if Reset = '1' then
                        startR <= '0';
                        startR2 <= '0';
                elsif rising_edge(Clk) then
                        startR <= Start;
                        startR2 <= StartR;
                end if;
        end process prStart;
```

```
        prDelay : process(Reset, Clk)
        begin
                if Reset = '1' then
                        counter <= 0;
                        sample <= to_float(0.0);
                        DataOut <= (others => '0');
                elsif rising_edge(Clk) then
                        if StartR = '1' and startR2 = '0' then
                                counter <= 1;
                                if lt(AnalogIn, to_float(0.0)) then -- less than
                                        sample <= to_float(0.0);
                                -- ge: greater than or equal to
                                elsif ge(AnalogIn, to_float(2**NBITS-1)) then
                                        sample <= to_float(2**NBITS-1);
                                else
                                        sample <= AnalogIn;
                                end if;
                        elsif counter = CYCLESDELAY then
                                DataOut <= std_logic_vector
                                        (to_unsigned(sample, NBITS));
                        elsif counter /= 0 then
                                counter <= counter + 1;
                        end if;
                end if;
        end process prDelay;
end Behavioral;
```

Código A.8: Modelo del ADC para simulación/emulación en *float*

A.2.3. Modelo del ADC para simulación/emulación en coma fija

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity AdcStdVector2StdVector is
generic(
        CYCLESDELAY : integer := 5;
        NBITS : integer := 12
);
port(
        Clk : in std_logic;
        Reset : in std_logic;
        Start : in std_logic;
        AnalogIn : in std_logic_vector(NBITS-1 downto 0);
        DataOut : out std_logic_vector(NBITS-1 downto 0)
);
end AdcStdVector2StdVector;

architecture Behavioral of AdcStdVector2StdVector is
```

```
        signal sample : std_logic_vector(NBITS-1 downto 0);
        signal counter : integer range 0 to CYCLESDELAY;
        signal startR, startR2 : std_logic;

begin

        prStart : process(Reset, Clk)
        begin
                if Reset = '1' then
                        startR <= '0';
                        startR2 <= '0';
                elsif rising_edge(Clk) then
                        startR <= Start;
                        startR2 <= StartR;
                end if;
        end process prStart;

        prDelay : process(Reset, Clk)
        begin
                if Reset = '1' then
                        counter <= 0;
                        sample <= (others => '0');
                        DataOut <= (others => '0');
                elsif rising_edge(Clk) then
                        if StartR = '1' and startR2 = '0' then
                                counter <= 1;
                        elsif counter = CYCLESDELAY then
                                DataOut <= sample;
                        elsif counter /= 0 then
                                counter <= counter + 1;
                        end if;
                end if;
        end process prDelay;

end Behavioral;
```

Código A.9: Modelo del ADC para simulación/emulación en coma fija

Apéndice B

Glosario de abreviaturas

AC	*Alternating Current*
ADC	*Analog-to-Digital Converter*
ASIC	*Application Specific Integrated Circuit*
BRAM	*Block RAM*
CCM	*Continuous Current Mode*
DC	*Direct Current*
DCM	*Discontinuous Current Mode* o *Digital Clock Manager*
DNLC	*Digital Non-Lineal Carrier*
DPWM	*Digital Pulse-Width Modulation*
DSP	*Digital Signal Processor*
FFT	*Fast Fourier Transform*
FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
HIL	*Hardware In-the-Loop*
ICON	*Integrated CONtroller*
ILA	*Integrated Logic Analyzer*
JTAG	*Joint Test Action Group*
LUT	*Look-Up Table*
LSB	*Low Significant Bit*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
PFC	*Power Factor Correction*
PWM	*Pulse-Width Modulation*
RAM	*Read Only Memory*
RMS	*Root Mean Square*
THD	*Total Harmonic Distortion*
VHDL	*VHSIC Hardware Description Language*
VHDL-AMS	*VHDL Analog and Mixed-Signal*
VHSIC	*Very-High-Speed Integrated Circuits*

Bibliografía

[1] Matlab, "www.mathworks.com," 2013.

[2] A. Prodic and D. Maksimovic, "Mixed-signal simulation of digitally controlled switching converters," in *Computers in Power Electronics, 2002. Proceedings. 2002 IEEE Workshop on*, pp. 100–105, jun. 2002.

[3] Spectre, "www.cadence.com," 2013.

[4] P. Zumel, M. García-Valderas, A. Lázaro, C. Loópez-Ongil, and A. Barrado, "Co-simulation psim-modelsim oriented to digitally controlled switching power converters," in *Control and Modeling for Power Electronics (COMPEL), 2010 IEEE 12th Workshop on*, pp. 1–7, jun. 2010.

[5] Modelsim, "www.mentor.com," 2013.

[6] A. de Castro, *Aplicación del Control Digital Basado en Hardware Específico para Convertidores de Potencia Conmutados*. PhD thesis, Universidad Politécnica de Madrid, 2003.

[7] P. Zumel, C. Fernandez, A. Lazaro, and A. Barrado, "Digital compensator design for dc-dc converters based on FPGA: an educational approach," in *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pp. 5439–5444, nov. 2006.

[8] L. Barragan, I. Urriza, D. Navarro, J. Artigas, J. Acero, and J. Burdio, "Comparing simulation alternatives of fpga-based controllers for switching converters," in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pp. 419–424, jun. 2007.

[9] A. de Castro, T. Riesgo, O. Garcia, and R. Prieto, "Comparing vhdl and vhdl-ams for modelling and simulation of power converters with digital control," in *XVIII Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 292–297, Nov. 2003.

[10] L. Laguna, R. Prieto, J. Oliver, and J. Cobos, "Top-down methodology employing hardware description languages (hdls) for designing digital control in power converters," in *Power Electronics Congress, 2008. CIEP 2008. 11th IEEE International*, pp. 133–137, aug. 2008.

[11] I. Urriza, L. Barragan, J. Artigas, J. Acero, D. Navarro, and J. Burdio, "Using mixed-signal simulation to design a digital power measurement system for induction heating home appliances," in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pp. 1447–1451, jun. 2007.

[12] O. Lucia, L. Barragan, J. Burdio, O. Jimenez, D. Navarro, and I. Urriza, "A versatile power electronics test-bench architecture applied to domestic induction heating," *Industrial Electronics, IEEE Transactions on*, vol. 58, pp. 998–1007, mar. 2011.

[13] O. Lucia, O. Jiménez, L. Barragán, I. Urriza, J. Burdío, and D. Navarro, "Real-time fpga-based hardware-in-the-loop development test-bench for multiple output power converters," in *Applied Power Electronics Conference and Exposition (APEC), 2010 Twenty-Fifth Annual IEEE*, pp. 309–314, feb. 2010.

[14] O. Lucia, I. Urriza, L. Barragán, D. Navarro, O. Jiménez, and J. Burdío, "Real-time fpga-based hardware-in-the-loop simulation test bench applied to multiple-output power converters," *Industry Applications, IEEE Transactions on*, vol. 47, pp. 853–860, mar.-apr. 2011.

[15] S. Karimi, A. Gaillard, P. Poure, and S. Saadate, "Fpga-based real-time power converter failure diagnosis for wind energy conversion systems," *Industrial Electronics, IEEE Transactions on*, vol. 55, pp. 4299–4308, dec. 2008.

[16] B. Lu, X. Wu, H. Figueroa, and A. Monti, "A low-cost real-time hardware-in-the-loop testing approach of power electronics controls," *Industrial Electronics, IEEE Transactions on*, vol. 54, pp. 919–931, apr. 2007.

[17] G. Constantinides, N. Nicolici, and A. Kinsman, "Numerical data representations for FPGA-Based scientific computing," *Design Test of Computers, IEEE*, vol. 28, pp. 8–17, jul.-aug. 2011.

[18] C. Dufour, V. Lapointe, J. Belanger, and S. Abourida, "Hardware-in-the-loop closed-loop experiments with an fpga-based permanent magnet synchronous motor drive system and a rapidly prototyped controller," in *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pp. 2152–2158, jul. 2008.

[19] A.-M. Lienhardt, G. Gateau, and T. Meynard, "Digital sliding-mode observer implementation using fpga," *Industrial Electronics, IEEE Transactions on*, vol. 54, pp. 1865–1875, aug. 2007.

[20] M. Matar and R. Iravani, "Fpga implementation of the power electronic converter model for real-time simulation of electromagnetic transients," *Power Delivery, IEEE Transactions on*, vol. 25, pp. 852–860, apr. 2010.

[21] A. Myaing and V. Dinavahi, "Fpga-based real-time emulation of power electronic systems with detailed representation of device characteristics," *Industrial Electronics, IEEE Transactions on*, vol. 58, pp. 358–368, jan. 2011.

[22] G. Parma and V. Dinavahi, "Real-time digital hardware simulation of power electronics and drives," *Power Delivery, IEEE Transactions on*, vol. 22, pp. 1235–1246, apr. 2007.

[23] A. Gebregergis and P. Pillay, "Implementation of fuel cell emulation on dsp and dspace controllers in the design of power electronic converters," *Industry Applications, IEEE Transactions on*, vol. 46, pp. 285–294, jan.-feb. 2010.

[24] A. Monti, E. Santi, R. Dougal, and M. Riva, "Rapid prototyping of digital controls for power electronics," *Power Electronics, IEEE Transactions on*, vol. 18, pp. 915–923, May 2003.

[25] E. I. W. Groups, "Vhdl2008 float_pkg (http://www.eda.org/fphdl)," 2010.

[26] SystemVision, "www.mentor.com," 2013.

[27] Questa Advanced Simulator, "www.mentor.com," 2011.

[28] R. Erickson and D. Maksimovic, *Fundamentals of power electronics*. Kluwer Academic, 2001.

[29] H. Peng, A. Prodic, E. Alarcon, and D. Maksimovic, "Modeling of quantization effects in digitally controlled dc-dc converters," *Power Electronics, IEEE Transactions on*, vol. 22, pp. 208–215, jan. 2007.

[30] A. Peterchev and S. Sanders, "Quantization resolution and limit cycling in digitally controlled PWM converters," *Power Electronics, IEEE Transactions on*, vol. 18, pp. 301–308, jan. 2003.

[31] A. de Castro, P. Zumel, O. Garcia, T. Riesgo, and J. Uceda, "Concurrent and simple digital controller of an ac/dc converter with power factor correction based on an fpga," *Power Electronics, IEEE Transactions on*, vol. 18, pp. 334–343, jan. 2003.

[32] B. Mather and D. Maksimović and, "A simple digital power-factor correction rectifier controller," *Power Electronics, IEEE Transactions on*, vol. 26, pp. 9–19, jan. 2011.

[33] M. Glasser, *Open Verification Methodology Cookbook*. Springer Dordrecht Heidelberg London New York, 2009.

[34] S. Buso, P. Mattavelli, L. Rossetto, and G. Spiazzi, "Simple digital control improving dynamic performance of power factor preregulators," *Power Electronics, IEEE Transactions on*, vol. 13, pp. 814 – 823, september 1998.

[35] J. Zhou, Z. Lu, Z. Lin, Y. Ren, Z. Qian, and Y. Wang, "A novel DSP controlled 2 kW PFC converter with a simple sampling algorithm," in *Applied Power Electronics Conference and Exposition, 2000. APEC 2000. Fifteenth Annual IEEE*, vol. 1, pp. 434–437, 2000.

[36] M. Fu and Q. Chen, "A DSP based controller for power factor correction (PFC) in a rectifier circuit," in *Applied Power Electronics Conference and Exposition, 2001. APEC 2001. Sixteenth Annual IEEE*, vol. 1, pp. 144–149, 2001.

[37] D. Diaz, O. Garcia, J. Oliver, P. Alou, F. Moreno, B. Duret, J. Cobos, F. Canales, and A. de Castro, "Digital control implementation to reduce the cost and improve the performance of the control stage of an industrial switch-mode power supply," in *Energy Conversion Congress and Exposition (ECCE), 2011 IEEE*, pp. 2930–2935, sept. 2011.

[38] C. Spiazzi, P. Mattavelli, and L. Rossetto, "Power factor preregulators with improved dynamic response," *Power Electronics, IEEE Transactions on*, vol. 12, pp. 343–349, mar 1997.

[39] S. Buso, P. Mattavelli, L. Rossetto, and G. Spiazzi, "Simple digital control improving dynamic performance of power factor preregulators," *Power Electronics, IEEE Transactions on*, vol. 13, no. 5, pp. 814–823, 1998.

[40] Y.-T. Feng, G.-L. Tsai, and Y.-Y. Tzou, "Digital control of a single-stage single-switch flyback pfc ac/dc converter with fast dynamic response," in *Power Electronics Specialists Conference, 2001. PESC. 2001 IEEE 32nd Annual*, vol. 2, pp. 1251–1256, 2001.

[41] A. Prodic, J. Chen, R. Erickson, and D. Maksimovic, "Digitally controlled low-harmonic rectifier having fast dynamic responses," in *Applied Power Electronics Conference and Exposition, 2002. APEC 2002. Seventeenth Annual IEEE*, vol. 1, pp. 476–482, mar. 2002.

[42] A. Prodic, J. Chen, D. Maksimovic, and R. Erickson, "Self-tuning digitally controlled low-harmonic rectifier having fast dynamic response," *Power Electronics, IEEE Transactions on*, vol. 18, pp. 420–428, jan. 2003.

[43] A. Prodic, "Compensator design and stability assessment for fast voltage loops of power factor correction rectifiers," *Power Electronics, IEEE Transactions on*, vol. 22, pp. 1719–1730, sept. 2007.

[44] S. Buso, S. Fasolo, and P. Mattavelli, "Uninterruptible power supply multiloop control employing digital predictive voltage and current regulators," *Industry Applications, IEEE Transactions on*, vol. 37, pp. 1846–1854, nov.-dec. 2001.

[45] P. Mattavelli, G. Spiazzi, and P. Tenti, "Predictive digital control of power factor preregulators with input voltage estimation using disturbance observers," *Power Electronics, IEEE Transactions on*, vol. 20, pp. 140–147, jan. 2005.

[46] W. Stefanutti, P. Mattavelli, G. Spiazzi, and P. Tenti, "Digital control of single-phase power factor preregulators based on current and voltage sensing at switch terminals," *Power Electronics, IEEE Transactions on*, vol. 21, pp. 1356–1363, sep. 2006.

[47] A. Pandey, B. Singh, and D. Kothari, "A novel dc bus voltage sensorless PFC rectifier with improved voltage dynamics," in *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, vol. 1, pp. 226–228, nov. 2002.

[48] K. Hwu, H. Chen, and Y. Yau, "Fully-digitalized implementation of pfc rectifier in ccm without adc," in *Applied Power Electronics Conference and Exposition (APEC), 2011 Twenty-Sixth Annual IEEE*, pp. 1464–1469, march 2011.

[49] K. Hwu, H. Chen, and Y. Yau, "Fully digitalized implementation of pfc rectifier in ccm without adc," *Power Electronics, IEEE Transactions on*, vol. 27, pp. 4021–4029, sept. 2012.

[50] A. Patel and M. Ferdowsi, "Current sensing for automotive electronics — a survey," *Vehicular Technology, IEEE Transactions on*, vol. 58, pp. 4108–4119, oct. 2009.

[51] S. Ziegler, R. Woodward, H.-C. Iu, and L. Borle, "Current sensing techniques: A review," *Sensors Journal, IEEE*, vol. 9, pp. 354–376, apr. 2009.

[52] P. Midya, P. Krein, and M. Greuel, "Sensorless current mode control-an observer-based technique for dc-dc converters," *Power Electronics, IEEE Transactions on*, vol. 16, pp. 522–526, jul. 2001.

[53] Y. Qiu, X. Chen, and H. Liu, "Digital average current-mode control using current estimation and capacitor charge balance principle for dc-dc converters operating in dcm," *Power Electronics, IEEE Transactions on*, vol. 25, pp. 1537–1545, jun. 2010.

[54] Z. Lukic, Z. Zhao, S. Ahsanuzzaman, and A. Prodic, "Self-tuning digital current estimator for low-power switching converters," in *Applied Power Electronics Conference and Exposition, 2008. APEC 2008. Twenty-Third Annual IEEE*, pp. 529–534, feb. 2008.

[55] Y.-S. Roh, Y.-J. Moon, J.-C. Gong, and C. Yoo, "Active power factor correction (PFC) circuit with resistor-free zero-current detection," *Power Electronics, IEEE Transactions on*, vol. 26, pp. 630–637, feb. 2011.

[56] F. Díaz, F. J. Azcondo, A. de Castro, and C. Brañas, "Controlador corrector de factor de potencia basado en técnicas de estimación de la corriente implementado en FPGA," in *Seminario Anual de AutomÃ¡tica, Electrónica Industrial e Instrumentación (SAAEI), 2008*, pp. 1–6, sep. 2008.

[57] F. J. Azcondo and A. de Castro, "Power factor correction controllers based on current rebuilding technique implemented on FPGA," in *International Exhibition & Conference for Power Electronics, Intelligent Motion, Power Quality, PCIM Europe 2008*, pp. 1–6, may. 2008.

[58] F. Díaz, F. J. Azcondo, A. de Castro, and C. Brañas, "Controlador corrector de factor de potencia basado en técnicas de estimación de la corriente implementado en FPGA," in *Seminario Anual de AutomÃ¡tica, Electrónica Industrial e Instrumentación (SAAEI), 2010*, pp. 1–6, jul. 2010.

[59] F. Azcondo, A. de Castro, V. Lopez, and O. Garcia, "Power factor correction without current sensor based on digital current rebuilding," *Power Electronics, IEEE Transactions on*, vol. 25, pp. 1527–1536, jun. 2010.

[60] V. Lopez, F. Azcondo, F. Diaz, and A. de Castro, "Autotuning digital controller for current sensorless power factor corrector stage in continuous conduction mode," in *Control and Modeling for Power Electronics (COMPEL), 2010 IEEE 12th Workshop on*, pp. 1–8, jun. 2010.

[61] V. Lopez-Martin, F. Azcondo, and A. de Castro, "Current error compensation for current-sensorless power factor corrector stage in continuous conduction mode," in *Control and Modeling for Power Electronics (COMPEL), 2012 IEEE 13th Workshop on*, pp. 1–8, jun. 2012.

[62] V. López, F. J. Azcondo, and A. de Castro, "High-resolution error compensation in continuous conduction mode power factor correction stage without current sensor," in *Power Electronics and Motion Control Conference (EPE-PEMC), 2012 ECCE Europe 15th International Conference on*, pp. 1–6, sep. 2012.

[63] M. Rodriguez, V. Lopez, F. Azcondo, J. Sebastian, and D. Maksimovic, "Average inductor current sensor for digitally controlled switched-mode power supplies," *Power Electronics, IEEE Transactions on*, vol. 27, pp. 3795–3806, aug. 2012.

[64] S. Sivakumar, K. Natarajan, and R. Gudelewicz, "Control of power factor correcting boost converter without instantaneous measurement of input current," *Power Electronics, IEEE Transactions on*, vol. 10, pp. 435–445, jul 1995.

[65] Y.-K. Lo, H.-J. Chiu, and S.-Y. Ou, "Constant-switching-frequency control of switch-mode rectifiers without current sensors," *Industrial Electronics, IEEE Transactions on*, vol. 47, pp. 1172–1174, oct. 2000.

[66] H.-C. Chen, "Single-loop current sensorless control for single-phase boost-type SMR," *Power Electronics, IEEE Transactions on*, vol. 24, pp. 163–171, jan. 2009.

[67] H.-C. Chen, C.-C. Lin, and J.-Y. Liao, "Modified single-loop current sensorless control for single-phase boost-type SMR with distorted input voltage," *Power Electronics, IEEE Transactions on*, vol. 26, pp. 1322–1328, may 2011.

[68] I. Merfert, "Analysis and application of a new control method for continuous-mode boost converters in power factor correction circuits," in *Power Electronics Specialists Conference, 1997. PESC '97 Record., 28th Annual IEEE*, vol. 1, pp. 96–102, jun. 1997.

[69] I. W. Merfert, "Stored-duty-ratio control for power factor correction," in *Applied Power Electronics Conference and Exposition, 1999. APEC '99. Fourteenth Annual*, vol. 2, pp. 1123–1129, mar. 1999.

[70] W. Zhang, G. Feng, Y.-F. Liu, and B. Wu, "A digital power factor correction (pfc) control strategy optimized for dsp," *Power Electronics, IEEE Transactions on*, vol. 19, pp. 1474–1485, nov. 2004.

[71] W. Zhang, Y.-F. Liu, and B. Wu, "A new duty cycle control strategy for power factor correction and FPGA implementation," *Power Electronics, IEEE Transactions on*, vol. 21, pp. 1745–1753, nov. 2006.

[72] B. Mather and D. Maksimovic, "Quantization effects and limit cycling in digitally controlled single-phase PFC rectifiers," in *Power Electronics Specialists Conference, 2008. PESC 2008. IEEE*, pp. 1297–1303, jun. 2008.